



LABORATORIUM  
TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS AHMAD DAHLAN



# PETUNJUK PRAKTIKUM

---

## GRAFIKA LANJUT

Penyusun:  
Adhi Prahara, S.Si., M.Cs.

2019

## KATA PENGANTAR

Puji syukur kami panjatkan kehadiran Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga petunjuk praktikum Grafika Lanjut dapat diselesaikan dengan lancar. Kami ucapkan terima kasih kepada berbagai pihak yang telah mendukung dalam penyusunan petunjuk praktikum ini.

Petunjuk praktikum Grafika Lanjut disusun untuk memberikan panduan dan kemudahan bagi mahasiswa dalam memahami materi grafika lanjut seperti shaders, pencahayaan, shading, tekstur, dan pemodelan.

Kami sadar bahwa dalam penyusunan petunjuk praktikum ini masih banyak kekurangan sehingga kritik dan saran sangat kami harapkan.

Yogyakarta, Agustus 2019

Penyusun

## DAFTAR PENYUSUN

Adhi Prahara, M.Cs.

## HALAMAN REVISI

Yang bertanda tangan di bawah ini:

Nama : Adhi Prahara, M.Cs.

NIK/NIY : 60150841

Jabatan : Dosen Pengampu Mata Kuliah Grafika Lanjut

Dengan ini menyatakan pelaksanaan Revisi Petunjuk Praktikum Grafika Lanjut untuk Program Studi Teknik Informatika telah dilaksanakan dengan penjelasan sebagai berikut:

No	Keterangan Revisi	Tanggal Revisi	Nomor Modul
1	a. Menyesuaikan kode di semua pertemuan praktikum ke C++ 11. b. Menambahkan cover depan dan belakang.	25 Juli 2018	PP/018/VII/R1
2	a. Mengganti cover depan dan belakang. b. Menambahkan halaman daftar penyusun. c. Menambahkan halaman revisi. d. Menambahkan halaman pernyataan. e. Menambahkan halaman visi dan misi prodi teknik informatika. f. Menambahkan halaman tata tertib praktikum. g. Menambahkan halaman daftar isi, tabel dan gambar. h. Menambahkan bagian alokasi waktu, total skor, dan indikator capaian di setiap pertemuan praktikum. i. Menambahkan bagian tugas disetiap pertemuan praktikum. j. Menambahkan halaman lembar jawaban pretest dan post test di setiap pertemuan praktikum.	23 Agustus 2019	PP/018/VII/R2

Yogyakarta, 23 Agustus 2019

Penyusun



**Adhi Prahara, M.Cs.**

NIY. 60150841

## HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Lisna Zahrotun, S.T., M.Cs.

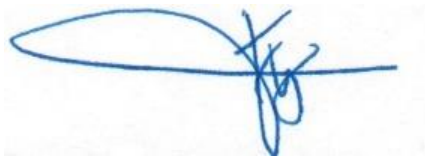
NIK/NIY : 60150773

Jabatan : Kepala Laboratorium Praktikum Teknik Informatika

Menerangkan dengan sesungguhnya bahwa Petunjuk Praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Gasal Tahun Akademik 2019/2020 di Laboratorium Praktikum Teknik Informatika, Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Ahmad Dahlan.

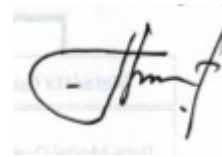
Yogyakarta, 23 Agustus 2019

Mengetahui,  
Ketua Kelompok Keilmuan Rekayasa Perangkat  
Lunak dan Data (RELATA)



**Drs., Tedy Setiadi, M.T.**  
NIY. 60030475

Kepala Laboratorium Praktikum Teknik  
Informatika



**Lisna Zahrotun, S.T., M.Cs.**  
NIY. 60150773

## VISI DAN MISI PRODI TEKNIK INFORMATIKA

### VISI

Menjadi Program Studi Informatika yang diakui secara internasional dan unggul dalam bidang Informatika serta berbasis nilai-nilai Islam.

### MISI

1. Menjalankan pendidikan sesuai dengan kompetensi bidang Informatika yang diakui nasional dan internasional
2. Meningkatkan penelitian dosen dan mahasiswa dalam bidang Informatika yang kreatif, inovatif dan tepat guna.
3. Meningkatkan kuantitas dan kualitas publikasi ilmiah tingkat nasional dan internasional
4. Melaksanakan dan meningkatkan kegiatan pengabdian masyarakat oleh dosen dan mahasiswa dalam bidang Informatika.
5. Menyelenggarakan aktivitas yang mendukung pengembangan program studi dengan melibatkan dosen dan mahasiswa.
6. Menyelenggarakan kerja sama dengan lembaga tingkat nasional dan internasional.
7. Menciptakan kehidupan Islami di lingkungan program studi.

## TATA TERTIB LABORATORIUM TEKNIK INFORMATIKA

### DOSEN/KOORDINATOR PRAKTIKUM

1. Dosen harus hadir saat praktikum minimal 15 menit di awal kegiatan praktikum dan menandatangani presensi kehadiran praktikum.
2. Dosen membuat modul praktikum, soal seleksi asisten, pre-test, post-test, dan responsi dengan berkoordinasi dengan asisten dan pengampu mata praktikum.
3. Dosen berkoordinasi dengan koordinator asisten praktikum untuk evaluasi praktikum setiap minggu.
4. Dosen menandatangani surat kontrak asisten praktikum dan koordinator asisten praktikum.
5. Dosen yang tidak hadir pada slot praktikum tertentu tanpa pemberitahuan selama 2 minggu berturut-turut mendapat teguran dari Kepala Laboratorium, apabila masih berlanjut 2 minggu berikutnya maka Kepala Laboratorium berhak mengganti koordinator praktikum pada slot tersebut.

### PRAKTIKAN

1. Praktikan harus hadir 15 menit sebelum kegiatan praktikum dimulai, dan dispensasi terlambat 15 menit dengan alasan yang jelas (kecuali asisten menentukan lain dan patokan jam adalah jam yang ada di Laboratorium, terlambat lebih dari 15 menit tidak boleh masuk praktikum & dianggap INHAL).
2. Praktikan yang tidak mengikuti praktikum dengan alasan apapun, wajib mengikuti INHAL, maksimal 4 kali praktikum dan jika lebih dari 4 kali maka praktikum dianggap GAGAL.
3. Praktikan harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
  - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
  - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
  - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
  - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Praktikan tidak boleh makan dan minum selama kegiatan praktikum berlangsung, harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di dalam laboratorium (tidak boleh membuang sampah sembarangan baik kertas, potongan kertas, bungkus permen baik di lantai karpet maupun di dalam ruang CPU).
5. Praktikan dilarang meninggalkan kegiatan praktikum tanpa seizin Asisten atau Laboran.
6. Praktikan harus meletakkan sepatu dan tas pada rak/loker yang telah disediakan.
7. Selama praktikum dilarang NGENET/NGE-GAME, kecuali mata praktikum yang membutuhkan atau menggunakan fasilitas Internet.
8. Praktikan dilarang melepas kabel jaringan atau kabel power praktikum tanpa sepengetahuan laboran
9. Praktikan harus memiliki FILE Petunjuk praktikum dan digunakan pada saat praktikum dan harus siap sebelum praktikum berlangsung.
10. Praktikan dilarang melakukan kecurangan seperti mencontek atau menyalin pekerjaan praktikan yang lain saat praktikum berlangsung atau post-test yang menjadi tugas praktikum.
11. Praktikan dilarang mengubah setting software/hardware komputer baik menambah atau mengurangi tanpa permintaan asisten atau laboran dan melakukan sesuatu yang dapat merugikan laboratorium atau praktikum lain.

12. Asisten, Koordinator Praktikum, Kepala laboratorium dan Laboran mempunyai hak untuk menegur, memperingatkan bahkan meminta praktikan keluar ruang praktikum apabila dirasa anda mengganggu praktikan lain atau tidak melaksanakan kegiatan praktikum sebagaimana mestinya dan atau tidak mematuhi aturan lab yang berlaku.
13. Pelanggaran terhadap salah satu atau lebih dari aturan diatas maka Nilai praktikum pada pertemuan tersebut dianggap 0 (NOL) dengan status INHAL.

### ASISTEN PRAKTIKUM

1. Asisten harus hadir 15 Menit sebelum praktikum dimulai (konfirmasi ke koordinator bila mengalami keterlambatan atau berhalangan hadir).
2. Asisten yang tidak bisa hadir WAJIB mencari pengganti, dan melaporkan kepada Koordinator Asisten.
3. Asisten harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
  - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
  - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
  - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
  - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Asisten harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di laboratorium, menegur atau mengingatkan jika ada praktikan yang tidak dapat menjaga kebersihan, ketertiban atau kesopanan.
5. Asisten harus dapat merapikan dan mengamankan presensi praktikum, Kartu Nilai serta tertib dalam memasukan/Input nilai secara Online/Offline.
6. Asisten harus dapat bertindak secara profesional sebagai seorang asisten praktikum dan dapat menjadi teladan bagi praktikan.
7. Asisten harus dapat memberikan penjelasan/pemahaman yang dibutuhkan oleh praktikan berkenaan dengan materi praktikum yang diasistensi sehingga praktikan dapat melaksanakan dan mengerjakan tugas praktikum dengan baik dan jelas.
8. Asisten tidak diperkenankan mengobrol sendiri apalagi sampai membuat gaduh.
9. Asisten dimohon mengkoordinasikan untuk meminta praktikan agar mematikan komputer untuk jadwal terakhir dan sudah dilakukan penilaian terhadap hasil kerja praktikan.
10. Asisten wajib untuk mematikan LCD Projector dan komputer asisten/praktikan apabila tidak digunakan.
11. Asisten tidak diperkenankan menggunakan akses internet selain untuk kegiatan praktikum, seperti Youtube/Game/Medsos/Streaming Film di komputer praktikan.

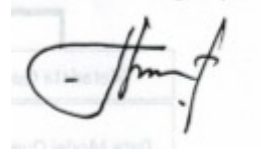
### LAIN-LAIN

1. Pada Saat Responsi Harus menggunakan Baju Kemeja untuk Laki-laki dan Perempuan untuk Praktikan dan Asisten.
2. Ketidakhadiran praktikum dengan alasan apapun dianggap INHAL.
3. Izin praktikum mengikuti aturan izin SIMERU/KULIAH.
4. Yang tidak berkepentingan dengan praktikum dilarang mengganggu praktikan atau membuat keributan/kegaduhan.
5. Penggunaan lab diluar jam praktikum maksimal sampai pukul 21.00 dengan menunjukkan surat ijin dari Kepala Laboratorium Prodi Teknik Informatika.



Yogyakarta, 23 Agustus 2019

Kepala Laboratorium Praktikum Teknik  
Informatika

A handwritten signature in black ink, appearing to read 'Lisna Zahrotun', is written over a faint, light blue grid background.

**Lisna Zahrotun, S.T., M.Cs.**

NIY. 60150773

## DAFTAR ISI

KATA PENGANTAR.....	1
DAFTAR PENYUSUN.....	2
HALAMAN REVISI.....	3
HALAMAN PERNYATAAN.....	4
VISI DAN MISI PRODI TEKNIK INFORMATIKA .....	5
TATA TERTIB LABORATORIUM TEKNIK INFORMATIKA.....	6
DAFTAR ISI.....	9
DAFTAR GAMBAR.....	10
DAFTAR TABEL.....	11
PRAKTIKUM 1: SHADERS.....	12
PRAKTIKUM 2: PENCAHAYAAN.....	19
PRAKTIKUM 3: SHADING.....	25
PRAKTIKUM 4: PEMETAAN TEKSTUR.....	35
PRAKTIKUM 5: TEKSTUR PROSEDURAL.....	42
PRAKTIKUM 6: MULTI-SAMPLE ANTI-ALIASING.....	50
PRAKTIKUM 7: RAY TRACING.....	56
PRAKTIKUM 8: MODEL PROSEDURAL.....	63
DAFTAR PUSTAKA.....	70

## DAFTAR GAMBAR

Gambar 1.1 Diagram Pipeline Grafis. ....	12
Gambar 1.2 Properties dari project. ....	13
Gambar 1.3 Mengisi linker pada properties. ....	14
Gambar 1.4 Tampilan hasil dari kode dasar. ....	14
Gambar 1.5 Menambahkan source file. ....	15
Gambar 1.6 Tampilan sebelum modifikasi shaders. ....	16
Gambar 1.7 Tampilan hasil modifikasi shaders. ....	17
Gambar 2.1 Tampilan model cahaya diffuse. ....	22
Gambar 2.2 Tampilan model cahaya ambient. ....	22
Gambar 2.3 Tampilan model cahaya specular. ....	23
Gambar 3.1 Model Goraud shading dan Phong shading ....	25
Gambar 3.2 Menambahkan source file, deklarasi header dan variable. ....	26
Gambar 3.3 Tampilan penerapan Goraud shading. ....	31
Gambar 3.4 Tampilan penerapan model Phong shading. ....	31
Gambar 3.5 Tampilan model flat shading. ....	32
Gambar 3.6 Tampilan model smooth shading. ....	33
Gambar 4.1 Skema Pemetaan Tekstur 2D ....	35
Gambar 4.2 Setting pada Visual Studio. ....	36
Gambar 4.3 Tampilan penerapan tekstur kayu. ....	39
Gambar 4.4 Tampilan penerapan tekstur batu bata. ....	39
Gambar 5.1 Contoh tekstur prosedural ....	42
Gambar 5.2 Menambahkan source file, deklarasi header dan variabel ....	43
Gambar 5.3 Tampilan penerapan tekstur procedural motif kayu ....	47
Gambar 5.4 Tampilan penerapan tekstur procedural rumput ....	48
Gambar 6.1 Tampilan penerapan MSAA. ....	54
Gambar 7.1 Ilustrasi ray tracing sederhana ....	57
Gambar 7.2 Tampilan deklarasi variable dan pengaturan ....	57
Gambar 7.3 Tampilan penerapan metode ray tracing pada bola. ....	61
Gambar 8.1 Jenis-jenis fractal sederhana ....	63
Gambar 8.2 Menambahkan source file, deklarasi header dan variable ....	64
Gambar 8.3 Tampilan fractal Mandelbrot ....	67

## DAFTAR TABEL

Tabel 6.1 Fungsi untuk menerapkan anti aliasing dengan OpenGL.....	50
--	----

## PRAKTIKUM 1: SHADERS

Pertemuan ke : 1

Total Alokasi Waktu : 90 menit

- Pre-Test : 15 menit
- Praktikum : 45 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

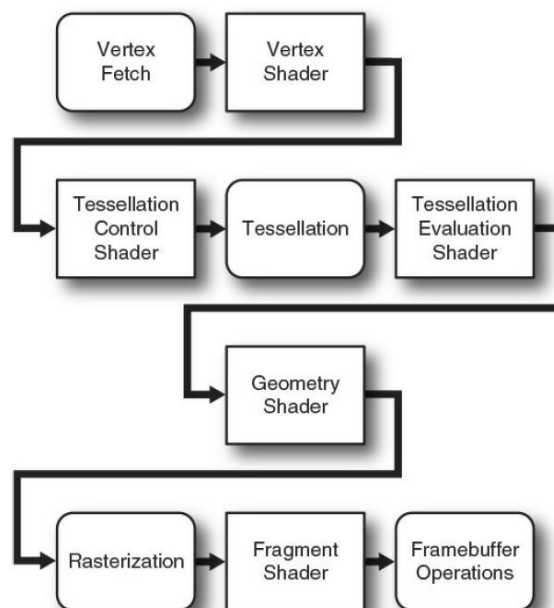
- Pre-Test : 20 %
- Praktikum : 30 %
- Post-Test : 50 %

### 1.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan shaders dengan API. Indikator ketercapaian diukur dengan kemampuan mahasiswa dalam menerapkan shaders dengan OpenGL API untuk menerapkan beberapa manipulasi vertex dan fragment shaders.

### 1.2. TEORI PENDUKUNG

Pada komputer grafis, kode atau program yang dibuat dengan OpenGL akan dikirim ke hardware grafis. Didalam hardware grafis, instruksi kode akan diproses secara parallel oleh GPU. GPU (*Graphics Processing Unit*) mempunyai banyak sekali prosesor yang disebut inti shaders (*shaders cores*) yang akan menjalankan program yang disebut dengan shaders. Pemrosesan grafis pada GPU akan mengikuti alur yang disebut dengan *graphics pipeline* seperti ditunjukkan pada Gambar 1.1.



Gambar 1.1 Diagram Pipeline Grafis.

Dari Gambar 1.1, terdapat 5 macam shaders yaitu *vertex shader*, *tessellation control shader*, *tessellation evaluation shader*, *geometry shader*, dan *fragment shader*. OpenGL shader ditulis dengan bahasa yang disebut OpenGL Shading Language atau disingkat GLSL. Bahasa ini aslinya ditulis dalam bahasa C tetapi sudah dimodifikasi untuk memudahkan penggunaannya. Dalam praktikum01 akan ditunjukkan bagaimana menuliskan kode pada shader terutama vertex shader dan fragment shader untuk memodifikasi obyek.

### 1.3. ALAT DAN BAHAN

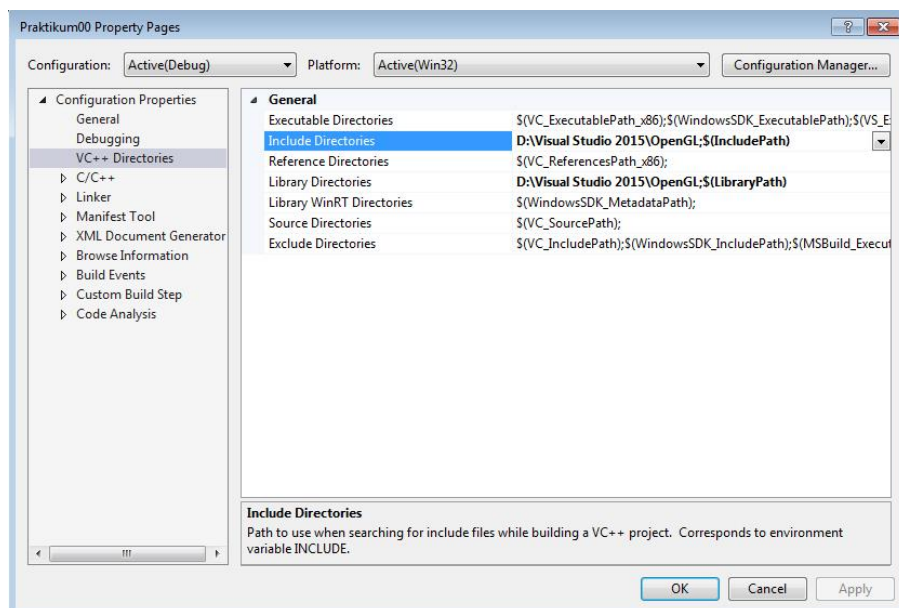
Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

### 1.4. LANGKAH PRAKTIKUM

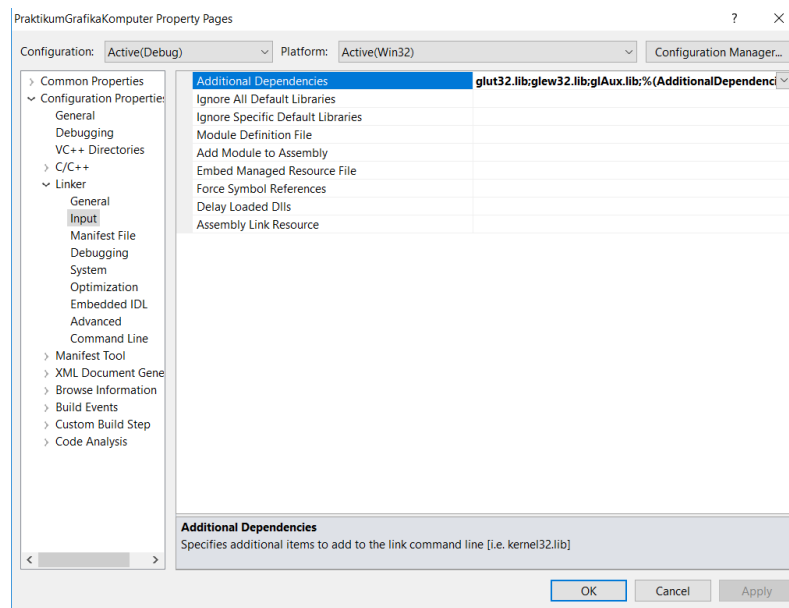
#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum01**.
2. Setting OpenGL library pada Visual Studio C/C++.
  - a. Download OpenGL (**opengl.rar**) dan kode dasar praktikum grafika lanjut (**praktikum00.cpp**) di e-learning dan masukkan file **praktikum00.cpp** ke **Source Files**.
  - b. Tentukan lokasi folder dimana OpenGL library diekstrak atau diletakkan pada hardisk.
  - c. Masuk ke project properties kemudian tambahkan **Include Directories** dan **Library Directories** OpenGL pada Visual Studio seperti ditunjukkan pada Gambar 1.2.



Gambar 1.2 Properties dari project.

- d. Isi **Linker** -> **Input** dengan **glut32.lib**, **glew32.lib**, dan **Glaux.lib** seperti ditunjukkan pada Gambar 1.3.



Gambar 1.3 Mengisi linker pada properties.

3. Jalankan kode dasar praktikum grafika lanjut untuk mengecek apakah pengaturan pada Visual Studio sudah benar.
4. Hasil tampilan jika kode dasar sudah berhasil dijalankan ditunjukkan pada Gambar 1.4.

NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA LANJ... — □ ×    NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA LANJ... — □ ×

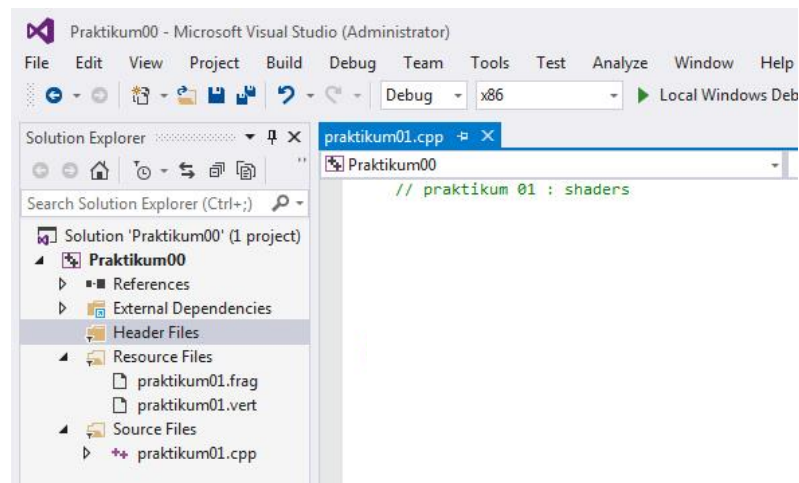


Gambar 1.4 Tampilan hasil dari kode dasar.

5. Geser kanan, kiri, atas, bawah, zoom in, dan zoom out untuk mengetahui apakah fungsi keyboard berjalan dengan benar.
6. Ubah nama file **praktikum00.cpp** menjadi **praktikum01.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 01 : SHADERS**

## PRAKTIKUM

1. Buat file dengan notepad kemudian simpan dengan nama **praktikum01.vert** untuk menulis vertex shaders dan **praktikum01.frag** untuk menulis fragment shaders lalu tambahkan ke project anda.
2. Hasil dari penambahan file diatas seperti ditunjukkan Gambar 1.5.



Gambar 1.5 Menambahkan source file.

3. Tambahkan variabel di praktikum01.cpp seperti kode di bawah ini.

```
GLuint vertexShaders, fragmentShaders, programShaders;
```

4. Tambahkan fungsi **readShaders()** untuk membaca file shaders berikut ke dalam **praktikum01.cpp**.

```
// fungsi membaca file shader
Char *readShaders(char *file)
{
    FILE *fptr;
    long length;
    char *buf;

    fopen_s(&fptr, file, "rb"); // buka file
    if (!fptr)
        return NULL;
    fseek(fptr, 0, SEEK_END);
    length = ftell(fptr);
    // buat buffer untuk menyimpan file
    buf = (char*)malloc(length + 1);
    fseek(fptr, 0, SEEK_SET);
    fread(buf, length, 1, fptr); // baca konten
    fclose(fptr); // tutup file
    buf[length] = 0;

    return buf;
}
```

5. Tambahkan fungsi **setShaders()** untuk membaca program shaders berikut ke dalam **praktikum01.cpp**.

```
// fungsi untuk membaca shaders
void setShaders()
{
    // baca vertex shaders
    char* vertex = readShaders("praktikum01.vert");
    // baca fragment shaders
    char* fragment = readShaders("praktikum01.frag");
    vertexShaders = glCreateShader(GL_VERTEX_SHADER);
    fragmentShaders = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(vertexShaders, 1, (const char**)&vertex, 0);
    glShaderSource(fragmentShaders, 1, (const char**)&fragment, 0);
    glCompileShader(vertexShaders);
    glCompileShader(fragmentShaders);
}
```



```

programShaders = glCreateProgram();
glAttachShader(programShaders, vertexShaders);
glAttachShader(programShaders, fragmentShaders);
glLinkProgram(programShaders);
glUseProgram(programShaders);

free(vertex);
free(fragment);
}

```

6. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi **glutSolidTeapot()** (teko) seperti dibawah ini.

```
glutSolidCube(1.0f); // menggambar obyek kubus
```

menjadi

```
glutSolidTeapot(1.0f); // menggambar obyek teapot
```

7. Tuliskan kode berikut ke dalam **praktikum01.vert** untuk memanipulasi vertex.

```

#version 150 core

void main(void)
{
    vec4 vertex = gl_Vertex;

    // scaling dengan 1.0 - ubah untuk scale up/down
    vertex.x = vertex.x * 1.0;
    vertex.y = vertex.y * 1.0;
    vertex.z = vertex.z * 1.0;

    gl_Position = gl_ModelViewProjectionMatrix * vertex;
}

```

8. Tuliskan kode berikut ke dalam **praktikum01.frag** untuk memanipulasi fragment.

```

#version 150 core

void main(void)
{
    // ubah warna asal dari hijau menjadi warna biru (0, 0, 1, 1)
    gl_FragColor = vec4(0.0, 0.0, 1.0, 1.0);
}

```

9. Jalankan program agar menghasilkan tampilan seperti pada Gambar 1.6.



Gambar 1.6 Tampilan sebelum modifikasi shaders.

10. Aktifkan shaders dengan memanggil fungsi **setShaders()** di fungsi **main()** tepat di bawah inisialisasi glew dan di atas fungsi **glutMainLoop()** seperti berikut.

```
...

// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
        glewGetString(GLEW_VERSION));

// aktifkan shaders
setShaders();

// looping
glutMainLoop();

...
```

11. Jalankan program agar menghasilkan tampilan seperti pada Gambar 1.7.



Gambar 1.7 Tampilan hasil modifikasi shaders.

Coba modifikasi parameter di dalam vertex shaders untuk mengubah-ubah ukuran dan lokasi obyek dan pada fragment shader untuk melihat perubahan warnanya.

### 1.5. TUGAS

1. Dengan mengedit vertex shader dan fragment shader, ubah obyek menjadi bola berwarna ungu, perkecil ukurannya menjadi setengah kalinya, dan translasikan ke pojok kanan atas!
2. Tuliskan kode untuk melakukan perubahan obyek tersebut di lembar jawab post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 1: SHADERS**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 2: PENCAHAYAAN

---

Pertemuan ke	: 2
Total Alokasi Waktu	: 90 menit
• Pre-Test	: 15 menit
• Praktikum	: 45 menit
• Post-Test	: 30 menit
Total Skor Penilaian	: 100%
• Pre-Test	: 20 %
• Praktikum	: 30 %
• Post-Test	: 50 %

---

### 2.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan model pencahayaan dengan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan model pencahayaan specular, diffuse, dan ambient.

### 2.2. TEORI PENDUKUNG

Pemodelan cahaya digunakan untuk membuat obyek tampak realistis seperti di dunia nyata. Pemodelan cahaya yang biasa dipakai yaitu pencahayaan model Phong. Dalam pencahayaan model Phong, bagaimana cahaya berinteraksi dengan permukaan yang digolongkan kedalam tiga kategori:

- **Cahaya sekitar / ambient light**  
Cahaya sekitar tidak berasal dari arah yang spesifik. Obyek menerima cahaya tidak langsung dari sumber cahaya tetapi berupa hasil pantulan tidak langsung dari sumber cahaya. Karakteristik obyek yang dikenai cahaya sekitar yaitu akan terang di seluruh permukaan di segala arah.
- **Cahaya tersebar / diffuse light**  
Cahaya tersebar adalah hasil interaksi sumber cahaya dengan permukaan yang menyebarkan cahaya karena permukaannya tidak rata atau kasar.
- **Cahaya biasa / specular light**  
Cahaya biasa merupakan hasil dari interaksi sumber cahaya dari arah tertentu terhadap permukaan benda. Permukaan yang terpancar cahaya akan terang dan yang tidak terpancar akan gelap tergantung dari sudut pandang terhadap posisi sumber cahaya dan obyek.

Dalam praktikum02 akan ditunjukkan bagaimana mengatur sumber cahaya dan interaksi cahaya terhadap obyek serta tentang sifat permukaan yang dikenai cahaya ambient, diffuse dan specular.

### 2.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL library.

## 2.4. LANGKAH PRAKTIKUM

### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum02**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti tahap persiapan pada **Praktikum 01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum02.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 02 : LIGHTING**

### PRAKTIKUM

1. Tambahkan variabel berikut ke dalam **praktikum02.cpp** untuk inisialisasi paramater pencahayaan.

```
// saklar untuk mematikan / menghidupkan pencahayaan
bool light0 = false;
bool light1 = false;
// material obyek
float matShininess[] = { 100.0 };
float matSurface[] = { 0.5, 0.5, 0.5, 0.0 };
float matSpecular[] = { 0.5, 0.5, 0.5, 0.0 };
// warna sumber cahaya
float lightColor0[] = { 0.0, 0.0, 1.0, 1.0 }; // biru
float lightColor1[] = { 1.0, 0.0, 0.0, 1.0 }; // merah
// posisi sumber cahaya
float lightPosition0[] = { 10.0, -5.0, 5.0, 1.0 };
float lightPosition1[] = { -10.0, -5.0, 5.0, 1.0 };
```

2. Tambahkan kode dibawah ini di dalam fungsi **display()** untuk menerapkan pencahayaan model diffuse dengan **GL\_DIFFUSE**.

```
// taruh semua obyek yang akan digambar di fungsi display()
void display()
{
    // bersihkan dan reset layar dan buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    // set warna dan posisi sumber cahaya
    // set posisi sumber cahaya
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition0);
    glLightfv(GL_LIGHT1, GL_POSITION, lightPosition1);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
    // perlakuan permukaan obyek
    // set material permukaan model
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, matSurface);
    glMaterialfv(GL_FRONT, GL_SPECULAR, matSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, matShininess);
    // matikan atau hidupkan sumber cahaya
    if (light0)
        glDisable(GL_LIGHT0);
    else
        glEnable(GL_LIGHT0);
    if (light1)
        glDisable(GL_LIGHT1);
    else
        glEnable(GL_LIGHT1);
}
```

```

// posisikan kamera pandang
gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ,
          0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek
drawObject();

glutSwapBuffers();
}

```

- Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi **glutSolidSphere()** (bola) seperti dibawah ini.

```
glutSolidCube(1.0f); // menggambar obyek kubus
```

menjadi

```
glutSolidSphere(1.0f, 50, 50); // menggambar obyek bola
```

- Tambahkan kode dibawah ini ke dalam fungsi **init()** untuk mengaktifkan pencahayaan.

```

... ..

// aktifkan pencahayaan dan warna material
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0); // sumber cahaya pertama (maksimal 8 sumber cahaya)
glEnable(GL_LIGHT1); // sumber cahaya kedua (maksimal 8 sumber cahaya)
glEnable(GL_COLOR_MATERIAL); // aktifkan perlakuan ke permukaan

... ..

```

- Tambahkan kode berikut ke dalam fungsi **keyboard()** untuk menon-aktifkan / mengaktifkan pencahayaan pada sumber cahaya 0 dan sumber cahaya 1 dengan menekan tombol F1 dan F2.

```

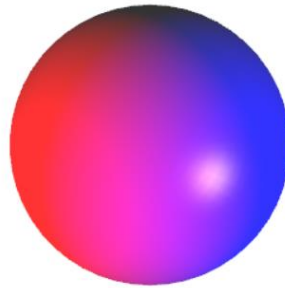
... ..

case GLUT_KEY_F1:
    // masukkan perintah disini bila tombol F1 ditekan
    light0 = !light0; // matikan / hidupkan sumber cahaya 0
    glutPostRedisplay();
    break;
case GLUT_KEY_F2:
    // masukkan perintah disini bila tombol F2 ditekan
    light1 = !light1; // matikan / hidupkan sumber cahaya 1
    glutPostRedisplay();
    break;

... ..

```

- Jalankan program agar menghasilkan tampilan seperti Gambar 2.1.



Gambar 2.1 Tampilan model cahaya diffuse.

7. Tampak pada Gambar 2.1 cahaya dari dua sumber cahaya tersebar di permukaan benda.
8. Praktikum ini menggunakan dua sumber cahaya. Gunakan tombol F1 dan F2 untuk mengaktifkan atau menon-aktifkan pencahayaan.
9. Untuk menerapkan model pencahayaan yang lain misalnya ambient, ubah **GL\_DIFFUSE** yang ada pada fungsi **display()** menjadi **GL\_AMBIENT** seperti dibawah ini.

```
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
```

menjadi

```
glLightfv(GL_LIGHT0, GL_AMBIENT, lightColor0);
glLightfv(GL_LIGHT1, GL_AMBIENT, lightColor1);
```

10. Jalankan program untuk menghasilkan tampilan pada Gambar 2.2.



Gambar 2.2 Tampilan model cahaya ambient.

11. Tampak pada Gambar 2.2 bahwa dua sumber cahaya ambient menghasilkan warna merah muda dari warna merah dan biru.
12. Untuk menerapkan model pencahayaan yang lain misalnya specular, ubah **GL\_AMBIENT** yang ada pada fungsi **display()** menjadi **GL\_SPECULAR** seperti dibawah ini.

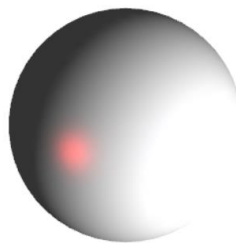
```
glLightfv(GL_LIGHT0, GL_AMBIENT, lightColor0);
glLightfv(GL_LIGHT1, GL_AMBIENT, lightColor1);
```

menjadi

```
glLightfv(GL_LIGHT0, GL_SPECULAR, lightColor0);
glLightfv(GL_LIGHT1, GL_SPECULAR, lightColor1);
```

13. Jalankan program untuk menghasilkan tampilan pada Gambar 2.3.

■ NAMA / NIM - PRAKTIKUM 02 GRAFIKA LANJUT — □ ×



Gambar 2.3 Tampilan model cahaya specular.

14. Tampak pada Gambar 2.3 bahwa dua sumber cahaya dipantulkan dengan sempurna pada permukaan benda (ubah posisi sumber cahaya untuk mendapatkan cahaya biru di sebelah kanan cahaya merah).
15. Amati perbedaan model pencahayaan dengan GL\_DIFFUSE, GL\_AMBIENT, dan GL\_SPECULAR.

## 2.5. TUGAS

1. Tambahkan sumber cahaya dari dua menjadi tiga sumber cahaya dengan posisi dan warna cahaya sebagai berikut:
 

```
// warna sumber cahaya
float lightColor0[] = { 0.0, 0.0, 1.0, 1.0 }; // biru
float lightColor1[] = { 1.0, 0.0, 0.0, 1.0 }; // merah
float lightColor2[] = { 0.0, 1.0, 0.0, 1.0 }; // hijau
// posisi sumber cahaya
float lightPosition0[] = { 10.0, -10.0, 10.0, 1.0 };
float lightPosition1[] = { -10.0, -10.0, 10.0, 1.0 };
float lightPosition2[] = { 0.0, 10.0, 10.0, 1.0 };
```
2. Terapkan model pencahayaan diffuse pada sumber cahaya 0, model pencahayaan ambient pada sumber cahaya 1, dan model pencahayaan specular pada sumber cahaya 2!
3. Tuliskan kode untuk melakukan model pencahayaan tersebut pada lembar jawaban post-test anda!



**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 2: PENCAHAYAAN**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 3: SHADING

---

Pertemuan ke	: 3
Total Alokasi Waktu	: 90 menit
• Pre-Test	: 15 menit
• Praktikum	: 45 menit
• Post-Test	: 30 menit
Total Skor Penilaian	: 100%
• Pre-Test	: 20 %
• Praktikum	: 30 %
• Post-Test	: 50 %

---

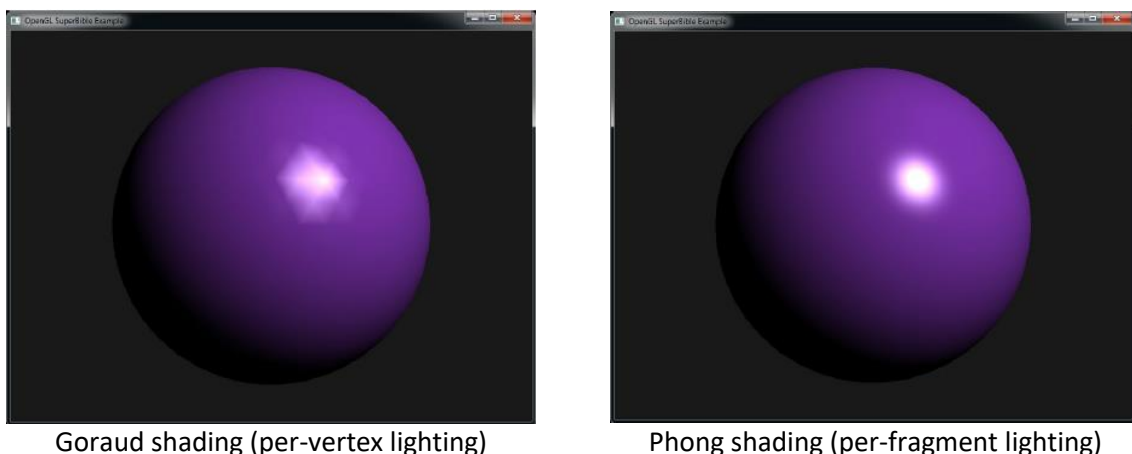
### 3.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan model shading dengan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan model shading flat, Goraud dan Phong.

### 3.2. TEORI PENDUKUNG

Selain dengan pecahayaan, untuk membuat obyek lebih realistis digunakan shading. Model shading yang biasa digunakan ada dua yaitu Goraud shading dan Phong shading. Goraud shading dilakukan dengan menghitung nilai cahaya per vertex kemudian menginterpolasi hasil antara nilai tersebut sebagai nilai shading. Kelemahan dari Goraud shading terlihat dari kilauan cahaya yang dihasilkan di permukaan benda yang ditunjukkan pada gambar bawah ini. Untuk menghilangkan efek tersebut akan dibutuhkan banyak vertex dalam geometri obyeknya.



Goraud shading (per-vertex lighting)

Phong shading (per-fragment lighting)

Gambar 3.1 Model Goraud shading dan Phong shading

Phong shading dilakukan dengan cara menginterpolasi permukaan normal antar vertex kemudian menggunakan hasilnya untuk menghitung nilai pencahayaan disetiap piksel bukan disetiap vertex seperti Goraud shading. Phong shading menampilkan hasil yang lebih baik dari Goraud shading

seperti ditunjukkan Gambar 3.1 dibawah ini. Selain Goraud dan Phong shading, pada praktikum03 akan ditunjukkan model shading yang lain yaitu flat shading dan interpolation shading.

### 3.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

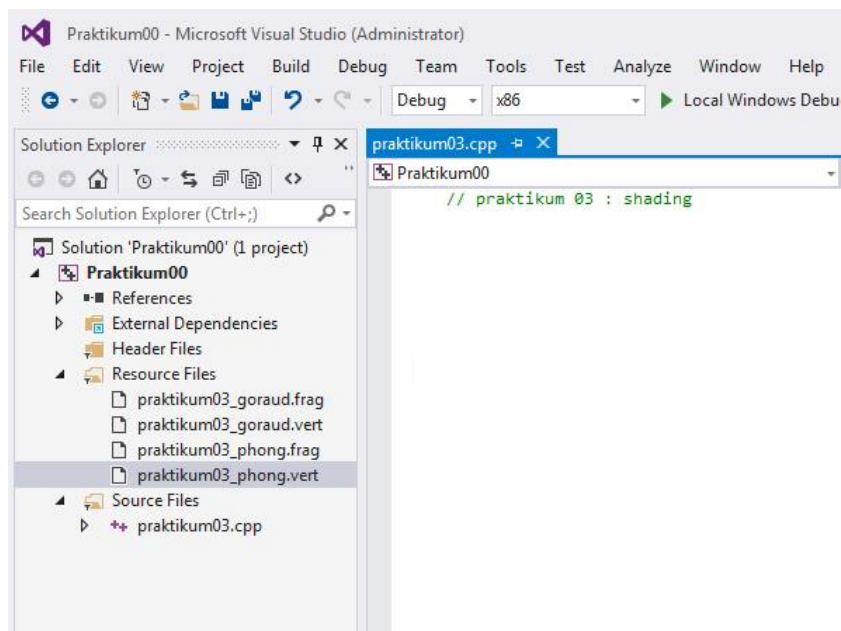
### 3.4. LANGKAH PRAKTIKUM

#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum03**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti tahap persiapan pada **Praktikum 01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum03.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 03 : SHADING**

#### PRAKTIKUM

1. Buat file dengan nama seperti dibawah ini dan tambahkan ke project anda:
  - a. **praktikum03\_goraud.vert** untuk menulis vertex shaders dari model Goraud shading
  - b. **praktikum03\_goraud.frag** untuk menulis fragment shaders dari model Goraud shading
  - c. **praktikum03\_phong.vert** untuk menulis vertex shaders dari model Phong shading
  - d. **praktikum03\_phong.frag** untuk menulis fragment shaders dari model Phong shading
2. Hasil dari penambahan file diatas seperti ditunjukkan Gambar 3.2.



Gambar 3.2 Menambahkan source file, deklarasi header dan variable.

3. Tuliskan kode berikut pada file **praktikum03\_goraud.vert** untuk vertex shaders Goraud shading.

```
// material dan pencahayaan
uniform vec3 lightPos = vec3(10.0, 10.0, 10.0);
uniform vec3 lightDiffuse = vec3(1.0, 1.0, 1.0);
```

```

uniform vec3 lightSpecular = vec3(1.0);
uniform float powerSpecular = 20.0;
uniform vec3 lightAmbient = vec3(1.0, 1.0, 1.0);

varying vec4 color;

void main(void)
{
    // hitung vektor posisi model
    vec3 P = vec3(gl_ModelViewMatrix * gl_Vertex);

    // hitung vektor normal
    vec3 N = normalize(gl_NormalMatrix * gl_Normal);

    // hitung vektor cahaya
    vec3 L = normalize(gl_LightSource[0].position.xyz - P.xyz);

    // hitung vektor pandang
    vec3 V = normalize(-P);

    // hitung vektor refleksi
    vec3 R = reflect(-L, N);

    // hitung properti ambient, diffuse dan specular
    vec4 ambient = gl_FrontLightProduct[0].ambient;
    vec4 diffuse = max(dot(N, L), 0.0) * gl_FrontLightProduct[0].diffuse;
    diffuse = clamp(diffuse, 0.0, 1.0);
    vec4 specular = pow(max(dot(R, V), 0.0), gl_FrontMaterial.shininess) *
        gl_FrontLightProduct[0].specular;
    specular = clamp(specular, 0.0, 1.0);

    // hitung warna untuk digunakan di fragment shaders
    color = gl_FrontLightModelProduct.sceneColor + ambient + diffuse + specular;

    // hitung posisi clipping model
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

```

4. Tuliskan kode berikut pada file **praktikum03\_goraud.frag** untuk fragment shaders Goraud shading.

```

varying vec4 color;

void main(void)
{
    // tulis hasil warna ke framebuffer
    gl_FragColor = color;
}

```

5. Tuliskan kode berikut pada file **praktikum03\_phong.vert** untuk vertex shaders Phong shading.

```

varying vec3 N;
varying vec3 L;
varying vec3 V;

void main(void)
{
    // hitung vektor posisi
    vec3 P = vec3(gl_ModelViewMatrix * gl_Vertex);

    // hitung vektor normal
    N = gl_NormalMatrix * gl_Normal;

```

```

// hitung vektor cahaya
L = gl_LightSource[0].position.xyz - P.xyz;

// hitung vektor pandang
V = -P;

// hitung posisi clipping model
gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

```

6. Tuliskan kode berikut pada file **praktikum03\_phong.frag** untuk fragment shaders Phong shading.

```

varying vec3 N;
varying vec3 L;
varying vec3 V;

void main(void)
{
    // normalisasi semua vektor
    vec3 tN = normalize(N);
    vec3 tL = normalize(L);
    vec3 tV = normalize(V);

    // hitung vektor refleksi
    vec3 tR = reflect(-tL, tN);

    // hitung komponen diffuse dan specular
    vec4 diffuse = max(dot(tN, tL), 0.0) * gl_FrontLightProduct[0].diffuse;
    vec4 specular = pow(max(dot(tR, tV), 0.0), gl_FrontMaterial.shininess) *
        gl_FrontLightProduct[0].specular;

    // hitung warna ke frame buffer
    gl_FragColor = gl_FrontLightModelProduct.sceneColor + diffuse + specular;
}

```

7. Tambahkan variabel berikut ke dalam **praktikum03.cpp** untuk inisialisasi parameter pencahayaan dan parameter shading.

```

// mematikan / menghidupkan pencahayaan
bool light0 = false;
//
float matShininess[] = { 20.0 }; // kekuatan specular cahaya
float matAmbient[] = { 1.0, 1.0, 1.0, 0.0 }; // material cahaya sekitar
float matDiffuse[] = { 1.0, 1.0, 1.0, 0.0 }; // material cahaya tersebar
float matSpecular[] = { 1.0, 1.0, 1.0, 0.0 }; // material cahaya biasa
// posisi sumber cahaya 1
float lightPosition0[] = { 10.0, 10.0, 10.0, 1.0 };

GLuint vertexShaders, fragmentShaders, programShaders;

```

8. Tambahkan fungsi untuk membaca file shaders berikut ke dalam **praktikum03.cpp**.

```

// fungsi membaca file shader
char* readShaders(char *file)
{
    FILE *fptr;
    long length;
    char *buf;

    fopen_s(&fptr, file, "rb"); // buka file
    if (!fptr)
        return NULL;
}

```

```

fseek(fp_ptr, 0, SEEK_END);
length = ftell(fp_ptr);
buf = (char*)malloc(length + 1); // buat buffer untuk menyimpan file
fseek(fp_ptr, 0, SEEK_SET);
fread(buf, length, 1, fp_ptr); // baca konten
fclose(fp_ptr); // tutup file
buf[length] = 0;

return buf;
}

```

9. Tambahkan fungsi untuk membaca program shaders untuk model **Gouraud shading** berikut ke dalam **praktikum03.cpp**.

```

// fungsi untuk me-load shaders
void setShaders()
{
    char* vertex = readShaders("praktikum03_gouraud.vert");
    char* fragment = readShaders("praktikum03_gouraud.frag");

    vertexShaders = glCreateShader(GL_VERTEX_SHADER);
    fragmentShaders = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(vertexShaders, 1, (const char**)&vertex, 0);
    glShaderSource(fragmentShaders, 1, (const char**)&fragment, 0);
    glCompileShader(vertexShaders);
    glCompileShader(fragmentShaders);

    programShaders = glCreateProgram();
    glAttachShader(programShaders, vertexShaders);
    glAttachShader(programShaders, fragmentShaders);
    glLinkProgram(programShaders);
    glUseProgram(programShaders);

    free(vertex);
    free(fragment);
}

```

10. Tuliskan program di bawah ini yang berguna untuk memanggil fungsi membaca program shaders, di dalam fungsi **main()**. Letakkan tepat di bawah inisialisasi glew dan diatas fungsi **glutMainLoop()**.

```

... ..

// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
        glewGetString(GLEW_VERSION));

// aktifkan shaders
setShaders();

// looping
glutMainLoop();

... ..

```

11. Tambahkan kode dibawah ini di dalam fungsi **display()** untuk menerapkan pencahayaan dan sifat permukaan untuk shading.

```

... ..

```

```
// set material dan posisi sumber cahaya
// set posisi sumber cahaya 0
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition0);

// set material permukaan model ambient
glMaterialfv(GL_FRONT, GL_AMBIENT, matAmbient);
// set material permukaan model diffuse
glMaterialfv(GL_FRONT, GL_DIFFUSE, matDiffuse);
// set material permukaan model specular
glMaterialfv(GL_FRONT, GL_SPECULAR, matSpecular);
// set material permukaan mengkilat
glMaterialfv(GL_FRONT, GL_SHININESS, matShininess);
// matikan atau hidupkan sumber cahaya
if (light0)
    glDisable(GL_LIGHT0);
else
    glEnable(GL_LIGHT0);

... ..
```

12. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi **glutSolidSphere()** (bola) seperti dibawah ini.

```
// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
glColor3f(0.0f, 1.0f, 0.0f);
glutSolidCube(1.0f); // menggambar obyek kubus
```

menjadi

```
// set warna obyek ke warna magenta (1, 0, 1)
glColor3f(1.0f, 0.0f, 1.0f);
glutSolidSphere(1.0f, 50.0f, 50.0f);
```

13. Tambahkan kode berikut ke dalam fungsi **keyboard()** untuk menon-aktifkan / mengaktifkan pencahayaan pada sumber cahaya 0 dengan menekan tombol F1.

```
... ..

case GLUT_KEY_F1:
    // masukkan perintah disini bila tombol F1 ditekan
    light0 = !light0;          // matikan / hidupkan sumber cahaya 0
    glutPostRedisplay();
    break;

... ..
```

14. Jalankan program untuk mendapatkan tampilan penerapan **Graud shading** pada Gambar 3.3.



Gambar 3.3 Tampilan penerapan Goraud shading.

15. Untuk menerapkan model **Phong shading**, ubah fungsi membaca Goraud shading pada fungsi **setShaders()** seperti berikut.

```
// baca vertex shaders
char* vertex = readShaders("praktikum03_goraud.vert");
// fragment shaders
char* fragment = readShaders("praktikum03_goraud.frag");
```

menjadi

```
// baca vertex shaders
char* vertex = readShaders("praktikum03_phong.vert");
// fragment shaders
char* fragment = readShaders("praktikum03_phong.frag");
```

16. Jalankan program untuk mendapatkan tampilan penerapan **Phong shading** pada Gambar 3.4.



Gambar 3.4 Tampilan penerapan model Phong shading.



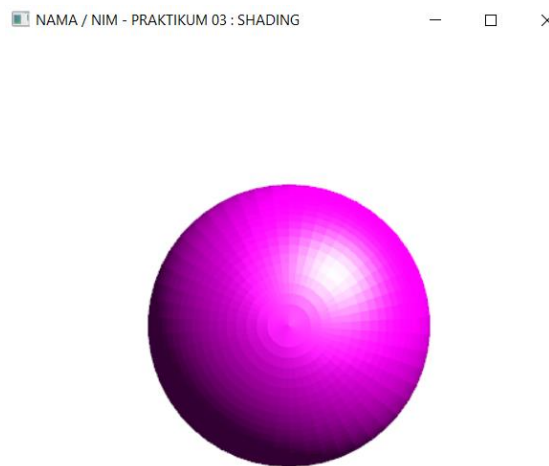
17. Perhatikan perbedaan antara Gouraud shaders dan Phong shaders.
18. Untuk menerapkan flat shading, **MATIKAN** fungsi **setShaders()** di fungsi **main()** dengan cara memberikan tanda **//** di depan fungsi **setShaders()** seperti berikut.

```
...
// matikan shaders
//setShaders();
...
```

19. Tambahkan kode berikut di dalam fungsi **display()** di bawah fungsi **glLookAt()** untuk menerapkan flat shading.

```
... ..
// terapkan shading
glShadeModel(GL_FLAT);      // flat shading
... ..
```

20. Jalankan program agar menghasilkan tampilan seperti pada Gambar 3.5.

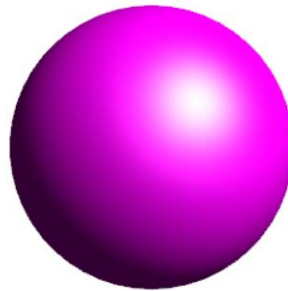


Gambar 3.5 Tampilan model flat shading.

21. Untuk menerapkan **interpolasi shading**, ubah **GL\_FLAT** menjadi **GL\_SMOOTH** seperti berikut di dalam fungsi **display()**.

```
... ..
// terapkan shading
glShadeModel(GL_SMOOTH);    // interpolative shading
... ..
```

22. Jalankan program agar menghasilkan tampilan seperti Gambar 3.6.



Gambar 3.6 Tampilan model smooth shading.

Perhatikan perbedaan antara beberapa shading di atas.

### 3.5. TUGAS

1. Ubah obyek menjadi teko/teapot berwarna ungu!
2. Terapkan model Phong shading pada teko tersebut!
3. Tuliskan kode untuk melakukan model Phong shading pada teko tersebut pada lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 3: SHADING**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 4: PEMETAAN TEKSTUR

Pertemuan ke : 4

Total Alokasi Waktu : 90 menit

- Pre-Test : 15 menit
- Praktikum : 45 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktikum : 30 %
- Post-Test : 50 %

### 4.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan pemetaan tekstur 2D pada obyek dengan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan pemetaan tekstur 2D pada obyek kubus dengan OpenGL.

### 4.2. TEORI PENDUKUNG

Tekstur merupakan bentuk terstruktur dari storage yang dapat diakses untuk dibaca atau ditulis oleh shaders. Biasanya digunakan untuk menyimpan data citra tekstur 2D. Tekstur dapat dipetakan dengan cara sebagai berikut:

- Menentukan koordinat 2D ( $s, t$ ) yang merupakan koordinat pada citra
- Normalisasi ke rentang  $[0, 1]$  yang kemudian disebut koordinat tekstur
- Memetakan koordinat tekstur ke setiap vertex di permukaan 3D
- Memetakan tekstur parametrik dengan mengubah nilai pikselnya sesuai yang diinginkan.

Skema pemetaan tekstur ditunjukkan Gambar 4.1 berikut. Pada praktikum04 akan ditunjukkan cara menerapkan tekstur 2D dengan menggunakan fungsi tekstur pada OpenGL.



Gambar 4.1 Skema Pemetaan Tekstur 2D

### 4.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.
4. Citra Tekstur.

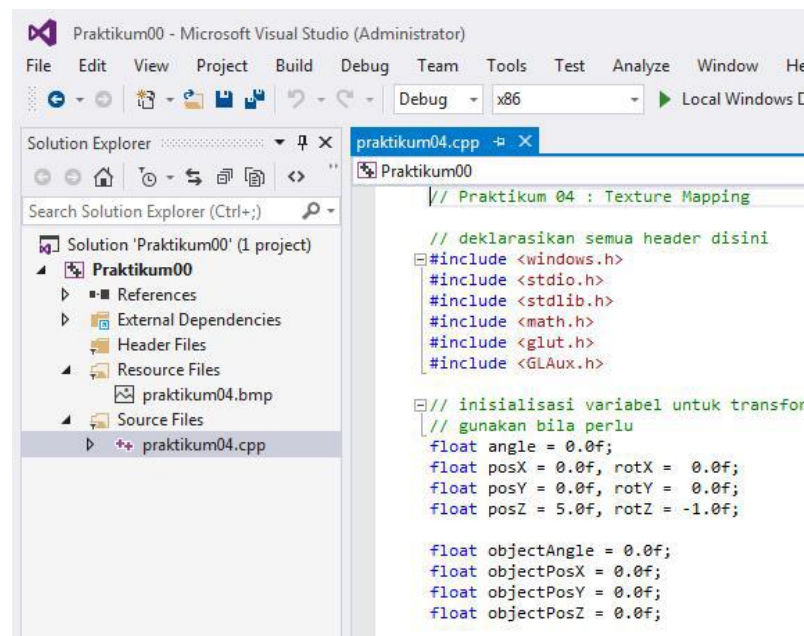
## 4.4. LANGKAH PRAKTIKUM

### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum04**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti tahap persiapan pada **Praktikum 01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum04.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 04 : PEMETAAN TEKSTUR**

### PRAKTIKUM

1. Download file citra tekstur dengan nama **praktikum04.bmp** dan letakkan pada project **praktikum04** anda seperti Gambar 4.2.



Gambar 4.2 Setting pada Visual Studio

2. Tambahkan fungsi untuk membaca citra tesktur berikut ke dalam **praktikum04.cpp**.

```
#pragma comment(lib, "legacy_stdio_definitions.lib")

// penyimpanan tekstur, dalam hal ini satu tekstur saja
GLuint texture[1];

// fungsi untuk membuka citra .bmp
AUX_RGBImageRec *loadBMP(char *filename)
{
    FILE *File = NULL;

    // cek apakah file tersedia
    fopen_s(&File, filename, "r");

    // bila tersedia lakukan prosedur membaca citra
    if (File)
    {

```

```

        fclose(File);
        size_t outSize;
        wchar_t wtext[256];
        mbstowcs_s(&outSize, wtext, filename, strlen(filename) + 1);
        LPWSTR filenamePtr = wtext;

        return auxDIBImageLoadW(filenamePtr);
    }
    // jika gagal kembalikan NULL
    return NULL;
}

```

3. Tuliskan fungsi **setTextures()** dibawah ini ke dalam **praktikum04.cpp**.

```

// fungsi untuk membuat tekstur dari citra tekstur
int setTextures()
{
    int status = FALSE;
    // buat alokasi memori untuk penyimpanan tekstur
    AUX_RGBImageRec *textureImage[1];
    // buat alokasi untuk teksturnya
    glGenTextures(1, texture);
    memset(textureImage, 0, sizeof(void *) * 1);

    // baca citra
    if (textureImage[0] = loadBMP("praktikum04.bmp"))
    {
        status = TRUE;

        // buat tekstur dari citra yang sudah dibaca tadi
        // dalam hal ini baca tekstur pertama
        glBindTexture(GL_TEXTURE_2D, texture[0]);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, textureImage[0]->sizeX,
                    textureImage[0]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE,
                    textureImage[0]->data);
        glGenerateMipmap(GL_TEXTURE_2D);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                        GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                        GL_LINEAR);
    }
    // jika tekstur sudah berhasil dibuat
    if (textureImage[0])
    {
        // bersihkan alokasi memori
        if (textureImage[0]->data)
            free(textureImage[0]->data);
        free(textureImage[0]);
    }
    return status;
}

```

4. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi kubus dengan sisi atas terbuka yang dibuat dengan **GL\_QUADS** seperti dibawah ini.

```

// fungsi untuk menggambar obyek
void drawObject()
{
    // obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()
    // fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek
    // lain saat diwarnai, ditransformasi dan sebagainya
    glPushMatrix();

```

```

// operasi transformasi rotasi obyek ke arah kanan-kiri
glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah
glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// sisi depan kotak
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, 1.0f);
glEnd();
//sisi belakang kotak
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
glEnd();
// sisi samping kanan kotak
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f,  1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f,  1.0f,  1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f,  1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glEnd();
// sisi samping kiri kotak
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glEnd();
// sisi bawah kotak
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glEnd();

glPopMatrix();

glPopMatrix();
}

```

5. Tambahkan kode berikut di dalam fungsi **init()** untuk mengaktifkan tekstur pada OpenGL.

```

... ..
// aktifkan tekstur
glEnable(GL_TEXTURE_2D);
... ..

```

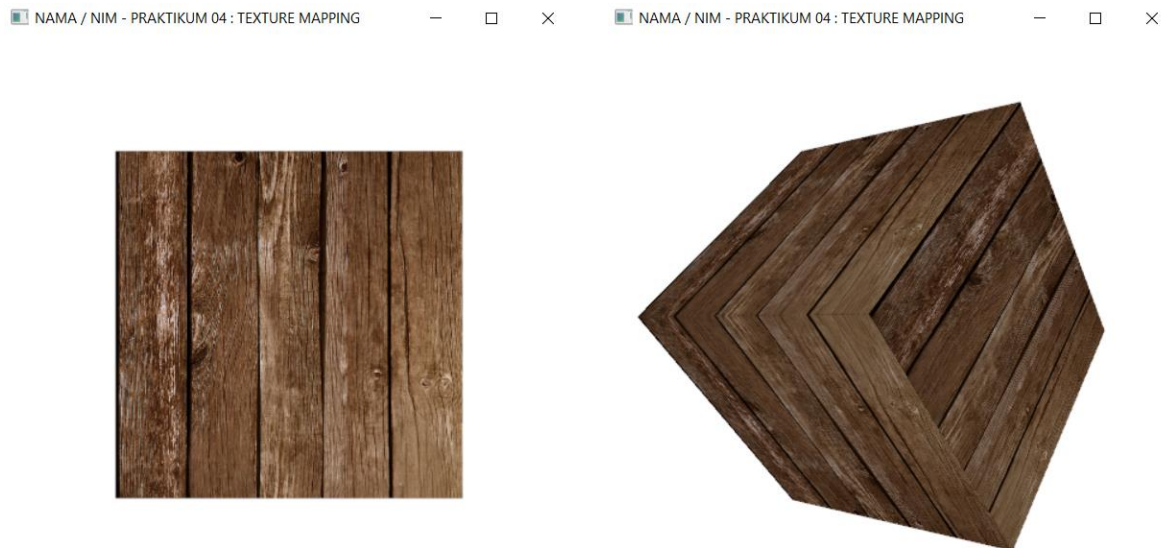
6. Tambahkan kode memanggil fungsi **setTexture()** di dalam fungsi **main()** di bawah inisialisasi glew dan di atas **glutMainLoop()** seperti di bawah ini.

```
... ..
// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
        glewGetString(GLEW_VERSION));

// set tekstur dengan citra yang sudah disediakan
int status = setTextures();

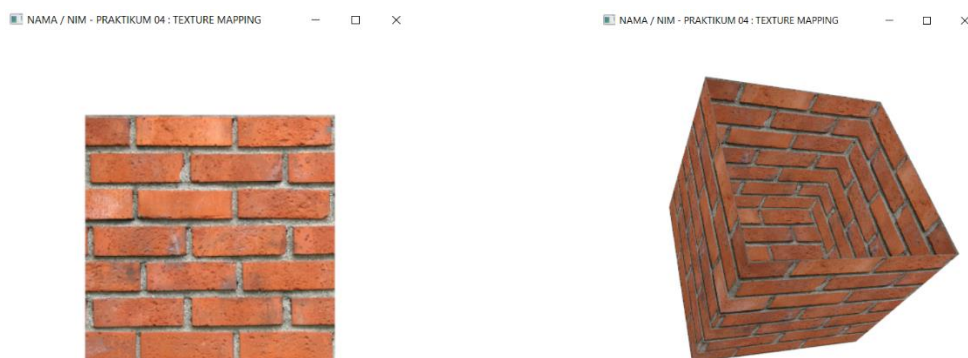
// looping
glutMainLoop();
... ..
```

7. Jalankan program agar menghasilkan tampilan pada Gambar 4.3.



Gambar 4.3 Tampilan penerapan tekstur kayu

8. Geser ke kiri, kanan, atas, bawah, zoom in, zoom out untuk melihat hasilnya.  
 9. Download citra tekstur kedua yaitu **praktikum04\_addon.bmp** dan terapkan hasilnya.  
 10. Jalankan program agar menghasilkan tampilan pada Gambar 4.4.



Gambar 4.4 Tampilan penerapan tekstur batu bata



#### 4.5. TUGAS

1. Terapkan dua tekstur yang berbeda (tekstur kayu dan tekstur batu bata) pada sisi kubus!
2. Tuliskan kode untuk melakukan pemetaan tekstur tersebut pada lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 4: PEMETAAN TEKSTUR**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 5: TEKSTUR PROSEDURAL

---

Pertemuan ke	: 5
Total Alokasi Waktu	: 90 menit
• Pre-Test	: 15 menit
• Praktikum	: 45 menit
• Post-Test	: 30 menit
Total Skor Penilaian	: 100%
• Pre-Test	: 20 %
• Praktikum	: 30 %
• Post-Test	: 50 %

---

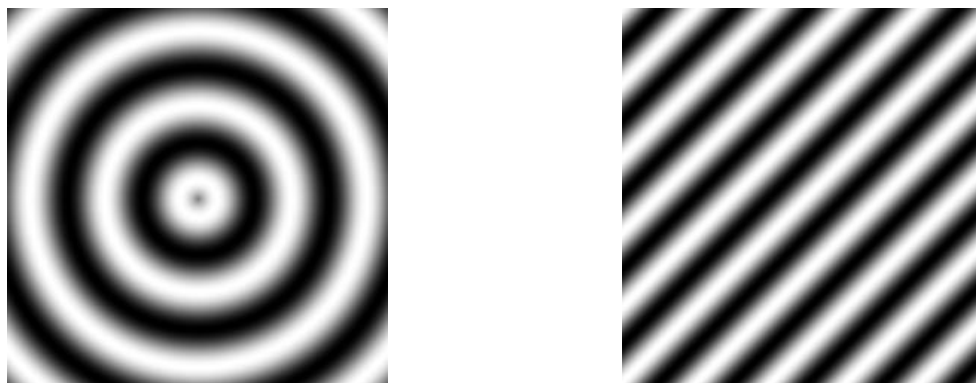
### 5.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan tekstur procedural pada obyek dengan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan tekstur procedural pada obyek kubus dengan OpenGL.

### 5.2. TEORI PENDUKUNG

Tekstur procedural merupakan tekstur yang dibuat dari sebuah fungsi yang menghitung koordinat tekstur. Tekstur yang bisa dibuat dengan fungsi prosedural seperti tekstur pola kayu dan marble yang dibuat dengan fungsi sinus harmonik yang diberi noise, tekstur langit, tekstur pasir dan sebagainya. Tekstur procedural mempunyai berbagai keuntungan seperti menghindari perhitungan transformasi tekstur dari koordinat citra 2D ke permukaan obyek, membutuhkan ruang simpan yang kecil, dan dapat divariasikan sesuai kebutuhan. Tekstur procedural juga mempunyai kelemahan yaitu apabila menggunakan fungsi yang kompleks maka pemrosesannya lambat. Contoh tekstur procedural ditunjukkan pada Gambar 5.1. Pada praktikum ini akan ditunjukkan penerapan tekstur procedural pada obyek kubus.



Gambar 5.1 Contoh tekstur prosedural

### 5.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

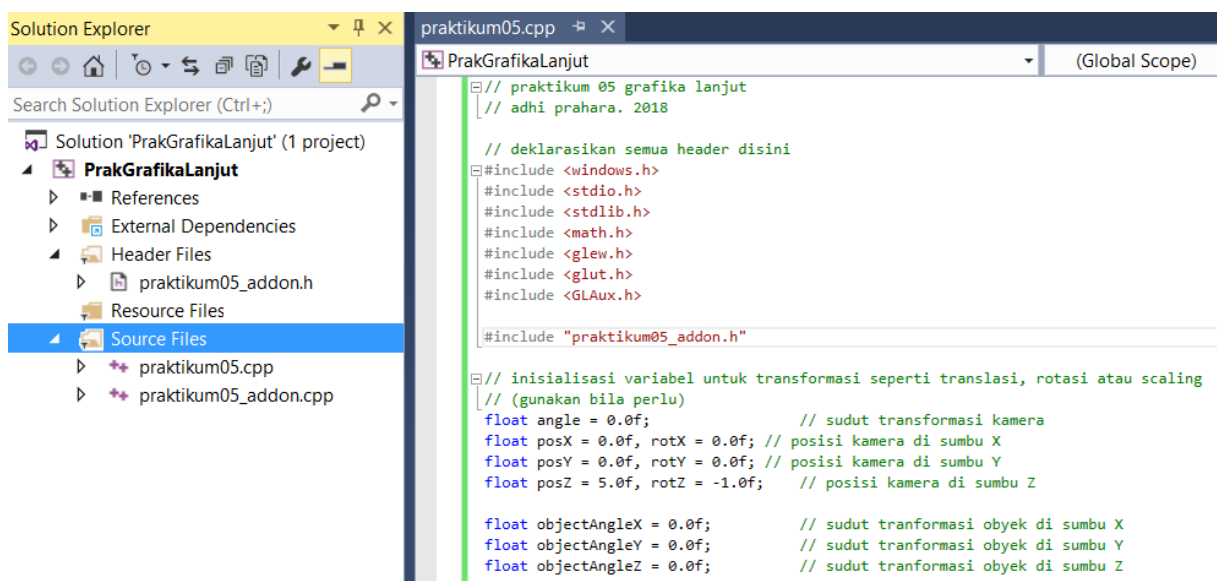
### 5.4. LANGKAH PRAKTIKUM

#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum05**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti pada tahap persiapan di praktikum01.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum05.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 05 : TESKTUR PROSEDURAL**

#### PRAKTIKUM

1. Download file tambahan di e-learning untuk membuat noise dengan fungsi Perlin noise dengan nama **praktikum05\_addon.cpp** dan **praktikum05\_addon.h**.
2. Tambahkan header **praktikum05\_addon.h** dan **praktikum05\_addon.cpp** seperti Gambar 5.2.



Gambar 5.2 Menambahkan source file, deklarasi header dan variabel

3. Tambahkan variabel dan fungsi untuk membuat noise berikut pada **praktikum05.cpp**.

```
#pragma comment(lib, "legacy_stdio_definitions.lib")

// penyimpanan tekstur, dalam hal ini satu tekstur saja
GLuint texture[1];

const int texWidth = 1024;
const int texHeight = 1024;
char texData[texWidth][texHeight][3]; // alokasi memori untuk tekstur 3D
double noiseData[texWidth][texHeight];

// fungsi untuk membangkitkan noise
void genNoise()
```

```

{
    for (int i = 0; i < texHeight; i++)
    {
        for (int j = 0; j < texWidth; j++)
            noiseData[i][j] = (rand() % 32768) / 32768.0;
    }
}

// fungsi untuk menghaluskan noise
double smoothNoise(double x, double y)
{
    // hitung nilai selisih fraksi dari x dan y
    double fractX = x - int(x);
    double fractY = y - int(y);

    // hitung jumlah langkah
    int x1 = (int(x) + texWidth) % texWidth;
    int y1 = (int(y) + texHeight) % texHeight;

    // hitung nilai ketetanggaan
    int x2 = (x1 + texWidth - 1) % texWidth;
    int y2 = (y1 + texHeight - 1) % texHeight;

    // haluskan noise
    double value = 0.0;
    value += fractX * fractY * noiseData[y1][x1];
    value += (1 - fractX) * fractY * noiseData[y1][x2];
    value += fractX * (1 - fractY) * noiseData[y2][x1];
    value += (1 - fractX) * (1 - fractY) * noiseData[y2][x2];

    return value;
}

// fungsi untuk memberi turbulensi pada noise
double turbNoise(double x, double y, double size)
{
    double value = 0.0, initialSize = size;

    while (size >= 1)
    {
        value += smoothNoise(x / size, y / size) * size;
        size /= 2.0;
    }

    return (128.0 * value / initialSize);
}

```

4. Tambahkan fungsi berikut pada **praktikum05.cpp** untuk membuat tekstur procedural motif kayu dengan harmonic sinus.

```

// fungsi untuk melakukan pemetaan tekstur
int setTextures()
{
    int status = FALSE;
    // buat alokasi untuk teksturnya
    glGenTextures(1, texture);

    genNoise();

    // buat tekstur dengan fungsi perlin
    float perlinData;
    for (int i = 0; i < texHeight; i++)
    {

```

```

for (int j = 0; j < texWidth; j++)
{
    // tekstur motif kayu dengan fungsi sinus
    double xyPeriod = 5.0; // banyaknya lingkaran
    double turbPower = 0.1; // dibuat twist
    double turbSize = 50.0; // turbulensi
    double xValue = (j - texWidth / 2) / double(texWidth);
    double yValue = (i - texHeight / 2) / double(texHeight);
    double distValue = sqrt(xValue * xValue + yValue *
        yValue) + 0.1 * turbNoise(j, i, turbSize) / 256.0;
    double sineValue = 128.0 * fabs(sin(2 * xyPeriod *
        distValue * 3.14159));
    texData[i][j][0] = (char)(80 + sineValue);
    texData[i][j][1] = (char)(30 + sineValue);
    texData[i][j][2] = 30;
}
}

status = TRUE;
// bind tekstur dari fungsi perlin tadi
glBindTexture(GL_TEXTURE_2D, texture[0]);
glTexImage2D(GL_TEXTURE_2D, 0, 3, texWidth, texHeight, 0, GL_RGB,
    GL_UNSIGNED_BYTE, texData);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

return status;
}

```

- Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi kubus dengan sisi atas terbuka yang dibuat dengan **GL\_QUADS** seperti dibawah ini.

```

// fungsi untuk menggambar obyek
void drawObject()
{
    // obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()
    // fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek
    // lain saat diwarnai, ditransformasi dan sebagainya
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // sisi depan kotak
    glBegin(GL_QUADS);
    // bind tekstur dari tekstur pertama
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
    glEnd();
    //sisi belakang kotak
    glBegin(GL_QUADS);
    // bind tekstur dari tekstur pertama
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

```

```

glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
glEnd();
// sisi samping kanan kotak
glBegin(GL_QUADS);
// bind tekstur dari tekstur pertama
glBindTexture(GL_TEXTURE_2D, texture[0]);
glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glEnd();
// sisi samping kiri kotak
glBegin(GL_QUADS);
// bind tekstur dari tekstur pertama
glBindTexture(GL_TEXTURE_2D, texture[0]);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glEnd();
// sisi bawah kotak
glBegin(GL_QUADS);
// bind tekstur dari tekstur pertama
glBindTexture(GL_TEXTURE_2D, texture[0]);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glEnd();

glPopMatrix();

glPopMatrix();
}

```

6. Tambahkan kode berikut ke dalam fungsi **init()** untuk mengaktifkan tekstur pada OpenGL dan melakukan inisialisasi tekstur.

```

... ..
// aktifkan tekstur
glEnable(GL_TEXTURE_2D);
... ..

```

7. Tambahkan kode berikut ke dalam fungsi **main()** tepat di bawah inisialisasi glew dan di atas fungsi **glutMainLoop()** untuk memanggil fungsi penerapan tekstur.

```

... ..
// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
        glewGetString(GLEW_VERSION));

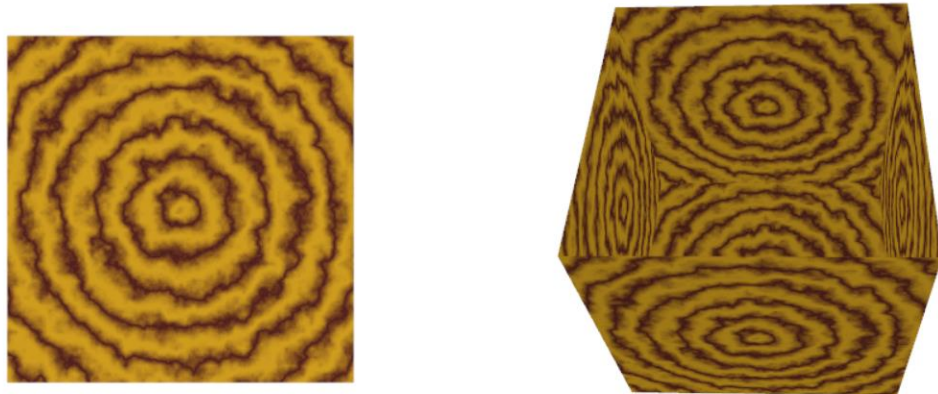
// set tekstur dengan citra prosedural
int status = setTextures();

// looping
glutMainLoop();
... ..

```

8. Jalankan program agar menghasilkan tampilan pada Gambar 5.3.

NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA LANJ... — □ × NAMA / NIM - KODE DASAR PRAKTIKUM GRAFIKA LANJ... — □ ×



Gambar 5.3 Tampilan penerapan tekstur procedural motif kayu

9. Untuk motif yang lain misalnya rumput, modifikasi fungsi **setTextures()** pada **praktikum05.cpp**.

```
// fungsi untuk melakukan pemetaan tekstur
int setTextures()
{
    int status = FALSE;
    // buat alokasi untuk teksturnya
    glGenTextures(1, texture);

    genNoise();

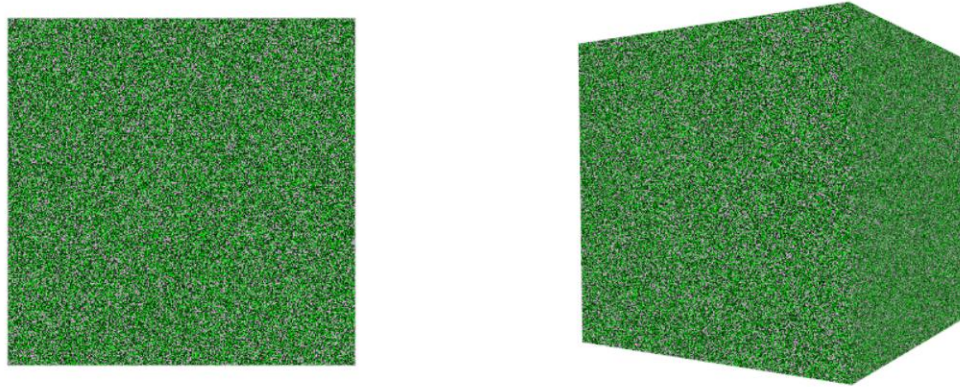
    // buat tekstur dengan fungsi perlin
    float perlinData;
    for (int i = 0; i < texHeight; i++)
    {
        for (int j = 0; j < texWidth; j++)
        {
            // tekstur rumput dengan perlin noise
            perlinData = perlinNoise2D(i, j, 0.1f, 0.9f, 8.0f);
            texData[i][j][0] = (char)perlinData / 64;
            texData[i][j][1] = (char)perlinData;
            texData[i][j][2] = (char)perlinData / 64;
        }
    }

    status = TRUE;
    // bind tekstur dari fungsi perlin tadi
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, texWidth, texHeight, 0, GL_RGB,
        GL_UNSIGNED_BYTE, texData);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

    return status;
}
```

10. Jalankan program untuk mendapatkan tampilan pada Gambar 5.4.





Gambar 5.4 Tampilan penerapan tekstur procedural rumput

11. Pada fungsi **PerlinNoise2D()** bila dimasukkan parameter tertentu akan membuat noise seperti tekstur rumput seperti gambar diatas. Warna hijau berasal dari manipulasi data RGB dengan data yang didapat dari Perlin noise di dalam fungsi **setTextures()**.

## 5.5. TUGAS

1. Buat tekstur awan di langit dari hasil modifikasi parameter di fungsi perlin noise di dalam fungsi **setTextures()**!
2. Tuliskan kode program untuk melakukan pembuatan tekstur procedural tersebut di lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 5: TEKSTUR PROSEDURAL**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 6: MULTI-SAMPLE ANTI-ALIASING

Pertemuan ke : 6

Total Alokasi Waktu : 90 menit

- Pre-Test : 15 menit
- Praktikum : 45 menit
- Post-Test : 30 menit

Total Skor Penilaian : 100%

- Pre-Test : 20 %
- Praktikum : 30 %
- Post-Test : 50 %

### 6.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan metode multi-sample anti aliasing pada obyek menggunakan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan metode multi-sample anti aliasing pada obyek kubus dengan OpenGL.

### 6.2. TEORI PENDUKUNG

Pada saat rendering sering terjadi aliasing. Aliasing biasanya berupa garis berundak pada tepian obyek yang disebabkan karena tidak semua pixel pada layar terpetakan dengan benar terhadap piksel teksturnya. Banyak solusi yang dapat digunakan untuk mengatasi aliasing yang disebut dengan anti-aliasing. Anti-aliasing ada beberapa cara seperti melakukan mip-mapping pada tekstur atau dengan menerapkan filter. Filter yang digunakan misalnya bilinear, trilinear maupun anisotropic. Untuk meningkatkan sample rate pada citra bisa dilakukan multi-sample anti-aliasing (MSAA). MSAA akan melakukan sample obyek primitif pada beberapa lokasi didalam piksel. Grafis modern sudah menggunakan teknik MSAA ini. Pada OpenGL, anti-aliasing dapat dilakukan tergantung pada obyek yang akan diterapkan anti aliasing. Pada praktikum06 akan ditunjukkan cara melakukan multi sampling anti aliasing dengan filter dan mipmapping.

Tabel 6.1 Fungsi untuk menerapkan anti aliasing dengan OpenGL

Obyek	Perintah OpenGL	Deskripsi
Garis	GL_LINE_SMOOTH	Anti-aliasing pada garis
Poligon	GL_POLYGON_SMOOTH	Anti-aliasing pada polygon
Tekstur	GL_NEAREST	Menerapkan filter nearest neighbor pada mip-map level dasar
	GL_LINEAR	Menerapkan filter linear pada mip-map level dasar
	GL_NEAREST_MIPMAP_NEAREST	Memilih level mip-map terdekat dan menerapkan filter nearest neighbor
	GL_NEAREST_MIPMAP_LINEAR	Melakukan interpolasi linear antara level mip-map dan menerapkan filter nearest neighbor
	GL_LINEAR_MIPMAP_NEAREST	Memilih level mip-map terdekat dan menerapkan filter linear
	GL_LINEAR_MIPMAP_LINEAR	Melakukan interpolasi linear antara level mip-map dan menerapkan filter linear disebut juga filter trilinear
MSAA	GL_MULTISAMPLE	Mengaktifkan multi-sample anti-aliasing

### 6.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

### 6.4. LANGKAH PRAKTIKUM

#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum06**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti pada tahap persiapan di **praktikum01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum06.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 06 : MSAA**

#### PRAKTIKUM

1. Tambahkan variabel berikut ke dalam **praktikum06.cpp** untuk inisialisasi parameter multi sample anti aliasing, VAO serta VBO untuk membuat kubus.

```
// parameter untuk multi sampling anti aliasing
bool flagMSAA = false;
GLuint framebuffer;
// parameter vertex array object (VAO) untuk kubus
// parameter vertex buffer object (VBO) untuk kubus
GLuint cubeVAO, cubeVBO;
// ukuran layar
int screenWidth = 480;
int screenHeight = 480;
```

2. Tambahkan fungsi untuk membuat multi sampel tekstur berikut ke dalam praktikum06.cpp.

```
// buat tekstur dengan multi sample
GLuint generateMultiSampleTexture(GLuint samples)
{
    GLuint texture;
    glGenTextures(1, &texture);
    glBindTexture(GL_TEXTURE_2D_MULTISAMPLE, texture);
    glTexImage2DMultisample(GL_TEXTURE_2D_MULTISAMPLE, samples, GL_RGB,
        screenWidth, screenHeight, GL_TRUE);
    glBindTexture(GL_TEXTURE_2D_MULTISAMPLE, 0);

    return texture;
}
```

3. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi seperti dibawah ini.

```
// fungsi untuk menggambar obyek
void drawObject()
{
    // obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()
    // fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek
    // lain saat diwarnai, ditransformasi dan sebagainya
    glPushMatrix();
```

```

// operasi transformasi rotasi obyek ke arah kanan-kiri
glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah
glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)
glColor3f(0.0f, 1.0f, 0.0f);

// aktifkan / matikan MSAA
if (flagMSAA)
    glDisable(GL_MULTISAMPLE);
else
    glEnable(GL_MULTISAMPLE);

float p = 1.0f;
GLfloat cubeVertices[] = {
    // sisi belakang
    -p, -p, -p,
    p, -p, -p,
    p, p, -p,
    -p, p, -p,
    // sisi depan
    -p, -p, p,
    p, -p, p,
    p, p, p,
    -p, p, p,
    // sisi kiri
    -p, p, p,
    -p, p, -p,
    -p, -p, -p,
    -p, -p, p,
    // sisi kanan
    p, p, p,
    p, p, -p,
    p, -p, -p,
    p, -p, p,
    // sisi bawah
    -p, -p, -p,
    p, -p, -p,
    p, -p, p,
    -p, -p, p,
    // sisi atas
    -p, p, -p,
    p, p, -p,
    p, p, p,
    -p, p, p
};

// buat VAO dan VBO untuk kubus
glGenVertexArrays(1, &cubeVAO);
glGenBuffers(1, &cubeVBO);
glBindVertexArray(cubeVAO);
glBindBuffer(GL_ARRAY_BUFFER, cubeVBO);
// masukkan data vertex kubus diatas ke buffer
glBufferData(GL_ARRAY_BUFFER, sizeof(cubeVertices), &cubeVertices,
             GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat),
                     (GLvoid*)0);
glBindVertexArray(0);

```

```

// gambar data kubus
glBindVertexArray(cubeVAO);
glDrawArrays(GL_QUADS, 0, 24);
glBindVertexArray(0);

// blit multi-sample buffer ke default frame buffer
glBindFramebuffer(GL_READ_FRAMEBUFFER, framebuffer);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, 0);
glBlitFramebuffer(0, 0, screenWidth, screenHeight, 0, 0, screenWidth,
    screenHeight, GL_COLOR_BUFFER_BIT, GL_NEAREST);

glPopMatrix();

glPopMatrix();
}

```

4. Tambahkan kode berikut di fungsi **display()** tepat di atas fungsi **glClear()** seperti berikut ini.

```

...

// bersihkan dan reset layar dan buffer
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
...

```

5. Tambahkan kode berikut ke dalam fungsi **init()**.

```

... ..
glEnable(GL_MULTISAMPLE); // mengaktifkan MSAA
... ..

```

6. Tambahkan kode berikut ke dalam fungsi **keyboard()** untuk mengaktifkan / mematikan fungsi MSAA dengan tombol F1.

```

... ..
case GLUT_KEY_F1:
    // masukkan perintah disini bila tombol F1 ditekan
    // dalam hal ini ubah switch dari MSAA
    flagMSAA = !flagMSAA;
    glutPostRedisplay(); // update obyek
    break;
... ..

```

7. Modifikasi **glutInitDisplayMode()** pada fungsi **main()** dengan kode berikut.

```

... ..
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH |
    GLUT_MULTISAMPLE);
... ..

```

8. Tambahkan kode berikut pada fungsi **main()** tepat di bawah inisialisasi **glew** dan di atas fungsi **glutMainLoop()**.

```

...
// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
        glewGetString(GLEW_VERSION));

// buat alokasi framebuffers

```

```

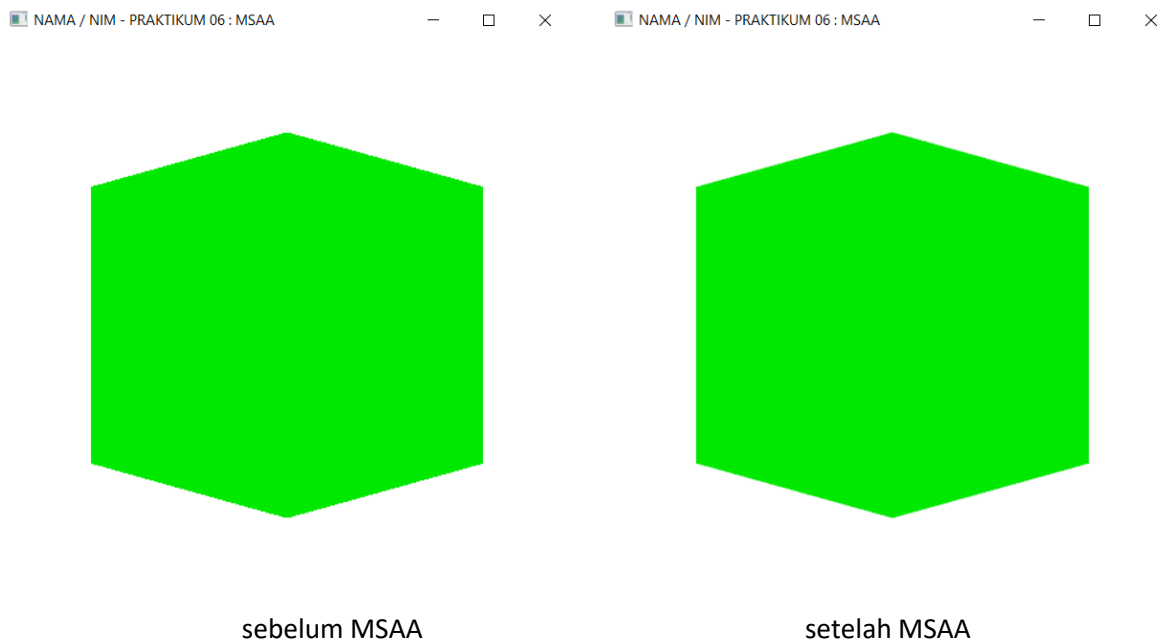
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);
// buat multi sample (dalam hal ini 4) warna tekstur
GLuint textureColorBufferMultiSampled = generateMultiSampleTexture(4);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
    GL_TEXTURE_2D_MULTISAMPLE, textureColorBufferMultiSampled, 0);

// buat alokasi render buffer untuk depth dan stencil
GLuint rbo;
glGenRenderbuffers(1, &rbo);
glBindRenderbuffer(GL_RENDERBUFFER, rbo);
glRenderbufferStorageMultisample(GL_RENDERBUFFER, 4, GL_DEPTH24_STENCIL8,
    screenWidth, screenHeight);
glBindRenderbuffer(GL_RENDERBUFFER, 0);
glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT,
    GL_RENDERBUFFER, rbo);
glBindFramebuffer(GL_FRAMEBUFFER, 0);

// looping
glutMainLoop();
...

```

9. Jalankan program agar menghasilkan tampilan sebelum menerapkan MSAA dan setelah menerapkan MSAA (zoom in untuk melihat perbedaannya).



Gambar 6.1 Tampilan penerapan MSAA

10. Terlihat bahwa multi sample anti aliasing dapat mengurangi jagged line (aliasing).

## 6.5. TUGAS

1. Buat 2 screenshot hasil sebelum dan sesudah penerapan multi sampling anti-aliasing (MSAA)!
2. Hasil (di zoom in biar kelihatan lebih jelas).
3. Tuliskan kode untuk menerapkan MSAA di lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 6: MULTI SAMPLE ANTI-ALIASING**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--



## PRAKTIKUM 7: RAY TRACING

---

Pertemuan ke	: 7
Total Alokasi Waktu	: 90 menit
• Pre-Test	: 15 menit
• Praktikum	: 45 menit
• Post-Test	: 30 menit
Total Skor Penilaian	: 100%
• Pre-Test	: 20 %
• Praktikum	: 30 %
• Post-Test	: 50 %

---

### 7.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan metode ray tracing dengan OpenGL.

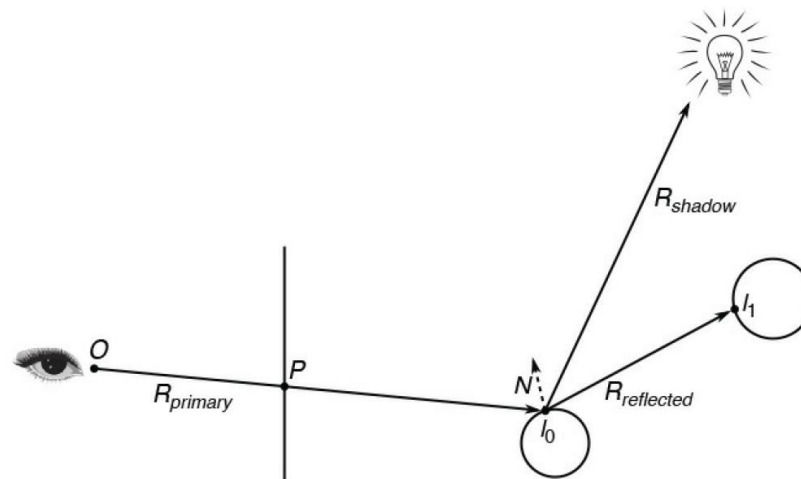
Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan metode ray tracing dengan OpenGL.

### 7.2. TEORI PENDUKUNG

Dalam melakukan rendering dapat dilakukan dengan dua cara yaitu rasterisasi dan ray casting. Ray casting dilakukan dengan menelusuri berkas cahaya yang jatuh atau memantul di obyek kembali ke sumber cahaya. Gambaran ray casting ditunjukkan pada Gambar 7.1. Prosedur melakukan ray casting adalah untuk setiap piksel pada layar dilakukan langkah sebagai berikut:

- Inisialisasikan warna pertama kali misalnya 0.
- Tentukan parameter sinar seperti titik asal sinar, jumlah sinar, arah sinar, dsb.
- Untuk setiap obyek pada layar, tentukan perpotongan terdekat antara sinar dengan obyek.
- Jika terjadi perpotongan maka untuk setiap sinar, bila sinar tidak berada di daerah bayangan obyek lain maka tambahkan pengaruh sinar pada permukaan warna obyek.
- Warna permukaan obyek adalah warna awal ditambah pengaruh sinar dikalikan factor pantulan.
- Faktor pantulan saat ini dihitung dengan mengalikan factor pantulan sebelumnya dengan property pantulan permukaan.
- Iterasi dilakukan sampai factor pantulan bernilai 0 atau kedalaman mencapai nilai yang ditentukan.

Perbedaan kelebihan rasterisasi dengan ray tracing adalah rasterisasi sulit mendapatkan tampilan yang bagus tapi cepat dalam pemrosesannya sedangkan ray tracing mudah mendapatkan tampilan yang bagus tapi perhitungannya lebih lama. Pada praktikum07 akan ditunjukkan peneraan metode ray tracing dengan OpenGL.



Gambar 7.1 Ilustrasi ray tracing sederhana

### 7.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

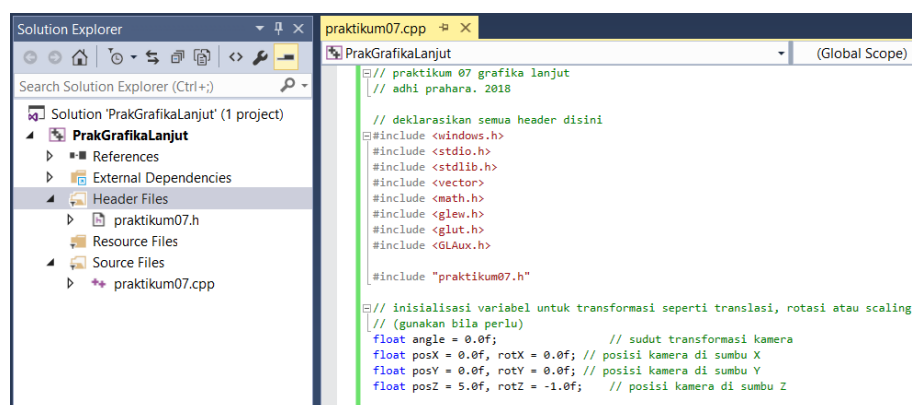
### 7.4. LANGKAH PRAKTIKUM

#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum07**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti pada tahap persiapan di **praktikum01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum07.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 07 : RAY TRACING**

#### PRAKTIKUM

1. Download file tambahan untuk fungsi tambahan untuk metode ray tracing dengan nama **praktikum07.h** dan tambahkan ke project anda.
2. Tambahkan header `#include <vector>` dan `#include "praktikum07.h"` untuk handle vektor array. Hasil dari penambahan file ditunjukkan Gambar 7.2.



Gambar 7.2 Tampilan deklarasi variable dan pengaturan

3. Tambahkan fungsi dan variable dibawah ini pada praktikum07.cpp untuk melakukan ray tracing.

```
// tentukan ukuran window OpenGL
#define screenWidth 480
#define screenHeight 480

// buat variable untuk menyimpan bola yang akan dibuat
std::vector<Sphere> spheres;

// lakukan ray tracing dengan obyek bola
Vec3f trace(
    const Vec3f &rayorig,
    const Vec3f &raydir,
    const std::vector<Sphere> &spheres,
    const int &depth)
{
    float tnear = INFINITY;
    const Sphere* sphere = NULL;

    // hitung perpotongan cahaya ini dengan bola yang ada dilayar
    for (unsigned i = 0; i < spheres.size(); ++i)
    {
        float t0 = INFINITY, t1 = INFINITY;
        if (spheres[i].intersect(rayorig, raydir, t0, t1))
        {
            if (t0 < 0) t0 = t1;
            if (t0 < tnear)
            {
                tnear = t0;
                sphere = &spheres[i];
            }
        }
    }

    // jika tidak ada perpotongan maka kembalikan warna background
    if (!sphere) return Vec3f(2);

    // warna permukaan obyek yang berpotongan dengan sinar
    Vec3f surfaceColor = 0;
    Vec3f phit = rayorig + raydir * tnear; // titik perpotongan
    Vec3f nhit = phit - sphere->center; // normal di titik perpotongan
    nhit.normalize(); // menormalisasikan arah normal
    // jika normal dan arah pandang tidak berlawanan satu sama lain
    // balikkan arah normal sehingga arah sekarang berada di dalam bola
    // membuat bagian dalam bola bernilai benar
    // tambahkan bias pada titik dimana akan di telusuri
    float bias = 1e-4;
    bool inside = false;
    if (raydir.dot(nhit) > 0)
        nhit = -nhit, inside = true;
    if ((sphere->transparency > 0 || sphere->reflection > 0) &&
        depth < MAX_RAY_DEPTH)
    {
        float facingratio = -raydir.dot(nhit);
        // ubah nilai mix untuk memberikan efek
        float fresneleffect = mix(pow(1 - facingratio, 3), 1, 0.1);
        // hitung arah refleksi sinar
        Vec3f reflldir = raydir - nhit * 2 * raydir.dot(nhit);
        reflldir.normalize();
        Vec3f reflection = trace(phit + nhit * bias, reflldir, spheres,
                                depth + 1);
        Vec3f refraction = 0;
        // jika bola nya transparan maka hitung sinar tembusnya
        if (sphere->transparency)
```

```

    {
        // cek apakah di dalam atau di luar permukaan?
        float ior = 1.1, eta = (inside) ? ior : 1 / ior;
        float cosi = -nhit.dot(raydir);
        float k = 1 - eta * eta * (1 - cosi * cosi);
        Vec3f refrdir = raydir * eta + nhit * (eta * cosi -
            sqrt(k));
        refrdir.normalize();
        refraction = trace(phit - nhit * bias, refrdir, spheres,
            depth + 1);
    }
    // hasilnya campuran dari pantulan dan tembusan cahaya
    surfaceColor = (reflection * fresneleffect + refraction *
        (1 - fresneleffect) * sphere->transparency) *
        sphere->surfaceColor;
}
else
{
    // bila obyek diffuse tidak perlu dilakukan ray tracing
    for (unsigned i = 0; i < spheres.size(); ++i)
    {
        if (spheres[i].emissionColor.x > 0)
        {
            // cahaya
            Vec3f transmission = 1;
            Vec3f lightDirection = spheres[i].center - phit;
            lightDirection.normalize();
            for (unsigned j = 0; j < spheres.size(); ++j)
            {
                if (i != j)
                {
                    float t0, t1;
                    if (spheres[j].intersect(phit + nhit *
                        bias, lightDirection, t0, t1))
                    {
                        transmission = 0;
                        break;
                    }
                }
            }
            surfaceColor += sphere->surfaceColor * transmission
                * max(float(0), nhit.dot(lightDirection)) *
                spheres[i].emissionColor;
        }
    }

    return surfaceColor + sphere->emissionColor;
}
}

```

4. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi bola seperti dibawah ini.

```

// fungsi untuk menggambar obyek
void drawObject()
{
    // obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()
    // fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek
    // lain saat diwarnai, ditransformasi dan sebagainya
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);
}

```

```

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah
glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

float invWidth = 1 / float(screenWidth);
float invHeight = 1 / float(screenHeight);
float fov = 45;
float aspectratio = screenWidth / float(screenHeight);
float angle = tan(M_PI * 0.5 * fov / 180.);

// telusuri sinar
for (unsigned y = 0; y < screenHeight; ++y)
{
    for (unsigned x = 0; x < screenWidth; ++x)
    {
        float xx = (2 * ((x + 0.5) * invWidth) - 1) * angle *
            aspectratio;
        float yy = (1 - 2 * ((y + 0.5) * invHeight)) * angle;
        Vec3f raydir(xx, yy, -1);
        raydir.normalize();
        Vec3f pixel = trace(Vec3f(0), raydir, spheres, 0);
        // tentukan warna setiap titik
        float r = min(float(1), pixel.x);
        float g = min(float(1), pixel.y);
        float b = min(float(1), pixel.z);
        // gambar titik warna permukaan hasil ray tracing
        glColor3f(r, g, b);
        glBegin(GL_POINTS);
        glVertex2f(-1.0f + 2 * (float)x / screenWidth, 1.0f - 2 *
            (float)y / screenHeight);
        glEnd();
    }
}

glPopMatrix();

glPopMatrix();
}

```

5. Ubah fungsi **display()** menjadi seperti dibawah ini.

```

// taruh semua obyek yang akan digambar di fungsi display()
void display()
{
    // bersihkan dan reset layar dan buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    // panggil fungsi untuk menggambar obyek
    drawObject();

    // tampilkan obyek ke layar
    // gunakan glFlush() bila memakai single buffer
    // gunakan glutSwapBuffers() bila memakai double buffer
    glutSwapBuffers();
}

```

6. Ubah fungsi **init()** menjadi seperti berikut:

```

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera
pandang di fungsi init()
void init(void)

```

```

{
    // inisialisasi warna latar belakang layar
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glEnable(GL_DEPTH_TEST);           // mengaktifkan depth buffer
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // set proyeksi ke orthogonal
    gluOrtho2D(0, screenWidth, 0, screenHeight);

    // posisi bola, radius bola, warna permukaan, tingkat pantulan,
    // transparansi, warna emisi
    // buat bola biasa
    spheres.push_back(Sphere(Vec3f(0.0, -10004, -20), 10000,
                               Vec3f(0.20, 0.20, 0.20), 0, 0.0));
    spheres.push_back(Sphere(Vec3f(0.0, 0, -30), 4,
                               Vec3f(1.00, 0.32, 0.36), 1, 0.5));
    spheres.push_back(Sphere(Vec3f(5.0, -1, -25), 2,
                               Vec3f(0.90, 0.89, 0.12), 1, 0.0));
    // buat bola cahaya
    spheres.push_back(Sphere(Vec3f(0.0, 20, -40), 3,
                               Vec3f(0.00, 0.00, 0.00), 0, 0.0, Vec3f(3)));
}

```

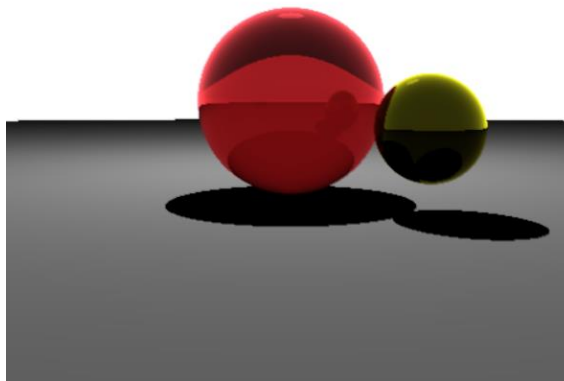
7. Non aktifkan fungsi **resize()** di fungsi **main()** menjadi seperti dibawah ini.

```

...
//glutReshapeFunc(reshape);    // reshape
...

```

8. Jalankan program agar menghasilkan tampilan seperti Gambar 7.3 (tunggu sampai gambar keluar karena rendering membutuhkan waktu agak lama tergantung spesifikasi komputer).



Gambar 7.3 Tampilan penerapan metode ray tracing pada bola

9. Pada Gambar 7.3, tampak dua buah bola memantulkan keadaan sekitarnya dengan baik dengan metode ray tracing.

## 7.5. TUGAS

1. Buat bola menjadi 4 buah dengan warna dan posisi yang berbeda-beda!
2. Tuliskan kode yang digunakan untuk melakukan hal tersebut di lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 7: RAY TRACING**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## PRAKTIKUM 8: MODEL PROSEDURAL

Pertemuan ke	: 8
Total Alokasi Waktu	: 90 menit
• Pre-Test	: 15 menit
• Praktikum	: 45 menit
• Post-Test	: 30 menit
Total Skor Penilaian	: 100%
• Pre-Test	: 20 %
• Praktikum	: 30 %
• Post-Test	: 50 %

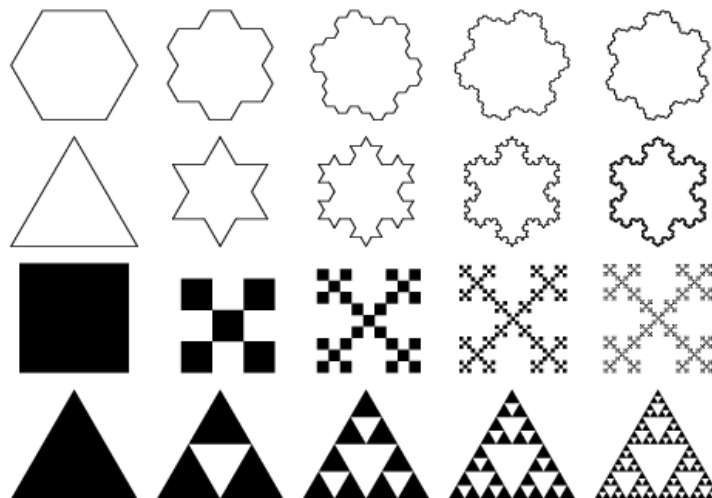
### 8.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan menerapkan model procedural terutama fractal dengan OpenGL.

Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan model fractal dengan OpenGL.

### 8.2. TEORI PENDUKUNG

Model procedural yang paling sering digunakan adalah fractal karena menampilkan tampilan yang menarik dengan perhitungan sederhana. Fraktal merupakan set matematika yang mempunyai pola yang berulang di setiap level ukuran. Jenis fractal ada banyak seperti Gosper island, Koch snowflake, Box fractal, Sierpinsky sieve, Barnsley's fern, Mandelbrot set, etc dan untuk fractal 3D ada Mandelbulb set. Beberapa contoh fractal ditunjukkan Gambar 8.1. Pada praktikum08 akan ditunjukkan penerapan fractal model Mandelbrot dan Julia set.



Gambar 8.1 Jenis-jenis fractal sederhana



### 8.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Visual Studio C++.
3. OpenGL Library.

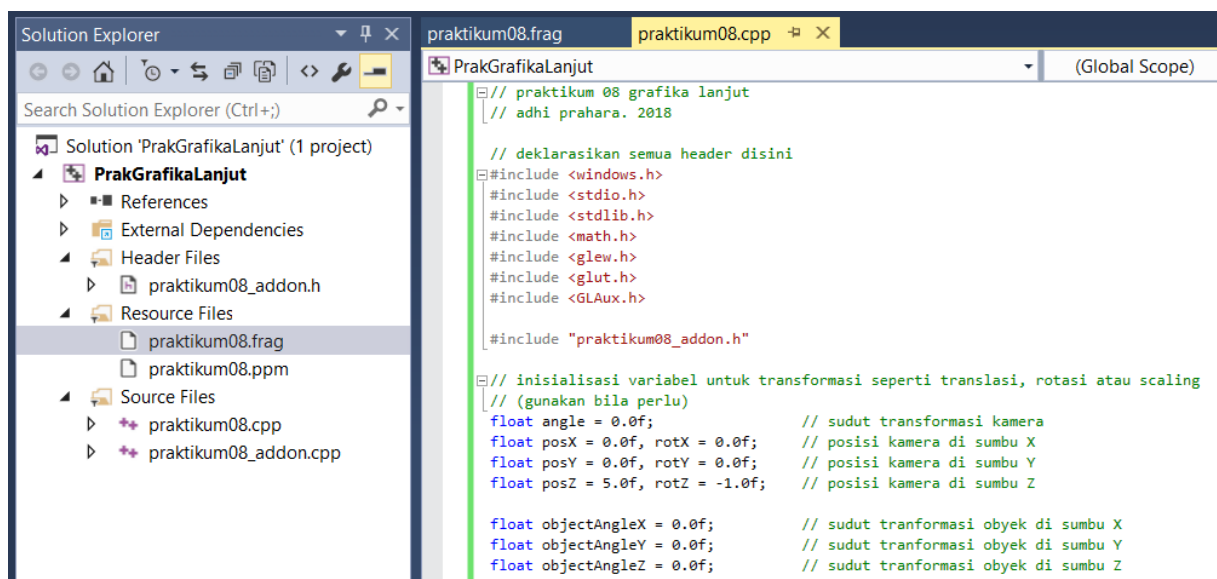
### 8.4. LANGKAH PRAKTIKUM

#### PERSIAPAN

1. Buka Visual Studio dan buat project baru dengan nama **praktikum08**.
2. Setting OpenGL library pada Visual Studio C/C++ seperti pada tahap persiapan di **praktikum01**.
3. Ubah nama file **praktikum00.cpp** menjadi **praktikum08.cpp** dan nama jendela menjadi **NAMA / NIM – PRAKTIKUM 08 : FRAKTAL MANDELBROT**

#### PRAKTIKUM

1. Download file tambahan untuk fungsi tambahan dengan nama **praktikum08\_addon.cpp**, **praktikum08\_addon.h** dan **praktikum08.ppm** dan tambahkan ke project anda.
2. Buat file dengan nama **praktikum08.frag** untuk menulis fragment shaders lalu tambahkan ke project anda seperti Gambar 8.2.



Gambar 8.2 Menambahkan source file, deklarasi header dan variable

3. Tambahkan variabel berikut ke dalam **praktikum08.cpp** untuk inisialisasi fraktal mandelbrot.

```
// parameter untuk menerapkan fractal mandelbrot
float scale = 2.2f;
float centerX = 0.7f;
float centerY = 0.0f;
int iterations = 70;
float zoomFactor = 0.025f;
GLuint programShaders;
```

4. Ubah obyek yang ada pada fungsi **drawObject()** dari **glutSolidCube()** (kubus) menjadi area kotak untuk menampilkan fraktal mandelbrot seperti dibawah ini.

```
// fungsi untuk menggambar obyek
```

```

void drawObject()
{
    // obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()
    // fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek
    // lain saat diwarnai, ditransformasi dan sebagainya
    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah kanan-kiri
    glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

    glPushMatrix();

    // operasi transformasi rotasi obyek ke arah atas-bawah
    glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

    // set titik tengah fraktal dan besarnya scaling
    setUniform2f(programShaders, "center", centerX, centerY);
    setUniform1f(programShaders, "scale", scale);

    // gambar wilayah kotak untuk fraktal
    glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex2f(-1, -1);
    glTexCoord2f(1, 0); glVertex2f(1, -1);
    glTexCoord2f(1, 1); glVertex2f(1, 1);
    glTexCoord2f(0, 1); glVertex2f(-1, 1);
    glEnd();

    glPopMatrix();

    glPopMatrix();
}

```

5. Modifikasi fungsi **display()** dengan menghapus / memberi double slash pada fungsi **glLookAt()**.

```

...
//gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ,
    0.0f, 1.0f, 0.0f);
...

```

6. Modifikasi fungsi **keyboard()** untuk menjelajahi fractal Mandelbrot.

```

// fungsi untuk mengatur masukan dari keyboard
// untuk arah kiri, kanan, atas, bawah, F1 dan F2
void keyboard(int key, int x, int y)
{
    // ambil ukuran layar
    int xres = glutGet(GLUT_WINDOW_WIDTH);
    int yres = glutGet(GLUT_WINDOW_HEIGHT);
    float fraction = 0.01f;

    switch (key)
    {
        // masukkan perintah disini bila tombol F1 ditekan
    case GLUT_KEY_F1:
        // naikan iterasi untuk fraktal
        // semakin banyak iterasi semakin banyak kita bisa zoom fraktal
        iterations += 10;
        printf("iterations: %d\n", iterations);
        setUniform1i(programShaders, "iter", iterations);
        glutPostRedisplay(); // update obyek
        break;
        // masukkan perintah disini bila tombol F2 ditekan
    case GLUT_KEY_F2:
        // turunkan iterasi untuk fraktal

```

```

        // semakin banyak iterasi semakin banyak kita bisa zoom fraktal
        iterations -= 10;
        if (iterations < 0) iterations = 0;
        printf("iterations: %d\n", iterations);
        setUniformli(programShaders, "iter", iterations);
        glutPostRedisplay();    // update obyek
        break;
        // masukkan perintah disini bila tombol kiri ditekan
    case GLUT_KEY_LEFT:
        // perintah geser obyek ke kiri sebanyak 1 derajat
        centerX -= fraction * scale / 2.0;
        glutPostRedisplay();    // update obyek
        break;
        // masukkan perintah disini bila tombol kanan ditekan
    case GLUT_KEY_RIGHT:
        // perintah geser obyek ke kanan sebanyak 1 derajat
        centerX += fraction * scale / 2.0;
        glutPostRedisplay();    // update obyek
        break;
        // masukkan perintah disini bila tombol atas ditekan
    case GLUT_KEY_UP:
        // perintah geser obyek ke atas sebanyak 1 derajat
        centerY += fraction * scale / 2.0;
        glutPostRedisplay();    // update obyek
        break;
        // masukkan perintah disini bila tombol bawah ditekan
    case GLUT_KEY_DOWN:
        // perintah geser obyek ke bawah sebanyak 1 derajat
        centerY -= fraction * scale / 2.0;
        glutPostRedisplay();    // update obyek
        break;
        // zoom in
    case GLUT_KEY_PAGE_UP:
        // masukkan perintah disini bila tombol PgUp ditekan
        scale *= 1 - zoomFactor * 2.0;
        glutPostRedisplay();    // update obyek
        break;
        // zoom out
    case GLUT_KEY_PAGE_DOWN:
        // masukkan perintah disini bila tombol PgDn ditekan
        scale *= 1 + zoomFactor * 2.0;
        glutPostRedisplay();    // update obyek
        break;
    }
}

```

7. Modifikasi fungsi **main()** untuk inisialisasi parameter fractal Mandelbrot. Perhatikan fungsi **init()** dan **glutReshapeFunc()** dinon-aktifkan / dihapus.

```

// program utama
int main(int argc, char** argv)
{
    ...

    // set ukuran jendela tampilan
    glutInitWindowSize(800, 600); // besarnya jendela dalam piksel
    glutInitWindowPosition(100, 100);

    ...

    // panggil fungsi init untuk inisialisasi awal
    //init();
}

```

```

...

//glutReshapeFunc(reshape);    // reshape

...

// inisialisasi glew
GLenum err = glewInit();
if (GLEW_OK != err)
    fprintf(stderr, "Error: %s\n", glewGetErrorString(err));
else
    fprintf(stdout, "Status: Using GLEW %s\n",
    glewGetString(GLEW_VERSION));

// membuka palet warna 1D untuk mewarnai fraktal
glBindTexture(GL_TEXTURE_1D, 1);
glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_WRAP_S, GL_REPEAT);
// ambil palet warna
void* image = loadImage("praktikum08.ppm", 0, 0);
glTexImage1D(GL_TEXTURE_1D, 0, 4, 256, 0, GL_BGRA, GL_UNSIGNED_BYTE,
    image);
free(image);
// aktifkan tekstur 1D
glEnable(GL_TEXTURE_1D);

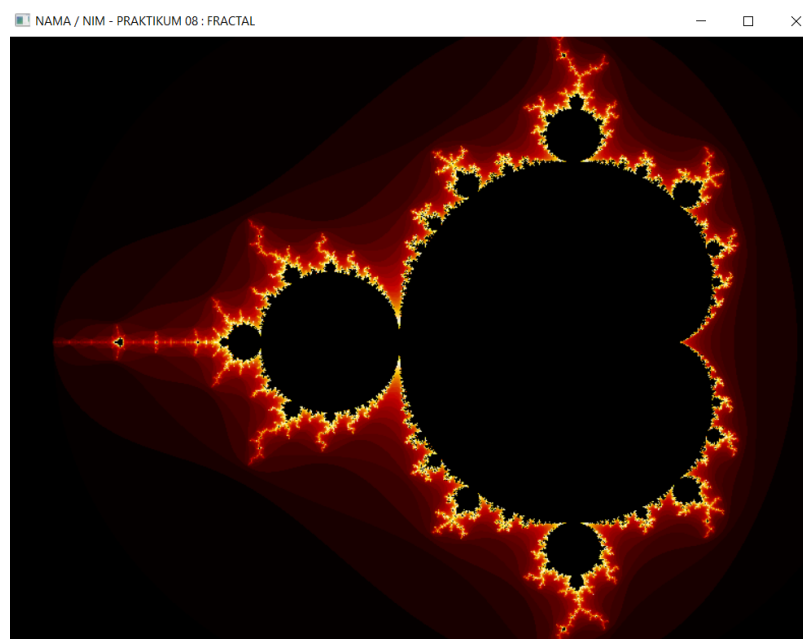
// me-load fragment shaders untuk mandelbrot fraktal
programShaders = setShader("praktikum08.frag");
setUniformli(programShaders, "iter", iterations);

// looping
glutMainLoop();

return 0;
}

```

8. Jalankan program agar menghasilkan tampilan seperti Gambar 8.3.



Gambar 8.3 Tampilan fractal Mandelbrot

9. Ubah parameter interaksi dengan menekan tombol F1 atau F2 dan lakukan zoom terhadap fractal dengan menekan tombol PgUp atau PgDn serta menggeser layar dengan menekan tanda panah pada keyboard.

### 8.5. TUGAS

1. Buat 4 screenshot dari hasil menjalankan program fractal Julia dengan posisi yang berbeda-beda!
2. Tuliskan kode program untuk menerapkan program fractal Julia yang anda buat di lembar jawaban post-test anda!

**LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM 8: MODEL PROSEDURAL**

<b>Nama :</b> <b>NIM :</b>	<b>Asisten:</b> <b>Paraf Asisten:</b>	<b>Tanggal:</b> <b>Nilai:</b>
-------------------------------	--	----------------------------------

--

## DAFTAR PUSTAKA

1. Edward Angel dan Dave Shreiner, 2011, Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL 6th Edition, Pearson Education
2. Donald Hearn dan M. Pauline Baker, 1997, Computer Graphics C Version 2nd Edition, Pearson Education
3. John F. Hughes, Andries Van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner dan Kurt Akeley, 2013, Computer Graphics Principles and Practice 3rd Edition, Addison-Wesley Professional
4. Dave Shreiner, Graham Sellers, John Kessenich and Bill Licea-Kane, 2013, OpenGL Programming Guide 8th Edition, Pearson Education
5. Edward Angel dan Dave Shreiner, 2014, Interactive Computer Graphics – A Top-Down Approach with WebGL 7th Edition, Pearson Education
6. R. Stuart Ferguson, 2014, Practical Algorithms for 3D Computer Graphics 2nd Edition, CRC Press: Taylor & Francis Group, LLC
7. Graham Sellers, Richard S. Wright, Jr, dan Nicholas Haemel. OpenGL SuperBible 7<sup>th</sup> Edition – Comprehensive Tutorial and Reference. Addison-Wesley. 2015.
8. <http://lodev.org/cgtutor/randomnoise.html>
9. [http://nuclear.mutantstargoat.com/articles/sdr\\_fract/](http://nuclear.mutantstargoat.com/articles/sdr_fract/)
10. <http://www.learnopengl.com/#!Advanced-OpenGL/Anti-aliasing>
11. [http://paulbourke.net/texture\\_colour/perlin/](http://paulbourke.net/texture_colour/perlin/)
12. <http://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/how-does-it-work>

